BAB I

PENDAHULUAN

1.1 Latar Belakang

Menurut laporan dari '*Global Status Report On Road Safety Status in 2018*' yang dirilis oleh *World Health Organization* (WHO), sekitar 1.350.000 orang meninggal dalam kecelakaan lalu lintas setiap tahunnya, 28% diantaranya meninggal akibat pengendara sepeda motor. Apalagi di beberapa daerah tertinggal, karena keterbatasan infrastruktur perkotaan dan kondisi ekonomi, sepeda motor telah menjadi alat transportasi utama, dan tingkat kematian lalu lintas jalan di daerah tersebut sekitar tiga kali lipat dari daerah maju (WHO, 2018).

Di Indonesia, menurut Korps Lalu Lintas Republik Indonesia (Korlantas Polri) mencatat jumlah kecelakaan dari tahun ke tahun terus meningkat, kecuali pada tahun 2020 dengan jumlah 100.028 kasus. Hal ini disebabkan pada tahun tersebut terjadi masa pandemi *Covid-19*. Akan tetapi, kasus kecelakan kembali naik sepanjang tahun 2022 sebanyak 139.258 kasus. Jumlah tersebut naik 31,16 persen dibandingkan pada tahun 2021 dengan 106.172 kejadian (Efliza & Suska, 2022). Salah satu penyebab terjadinya kecelakaan tersebut diakibatkan karena kurangnya perhatian masyarakat dalam mematuhi peraturan lalu lintas khususnya pengendara sepeda motor yang merupakan salah satu transportasi terbanyak di Indonesia.

Menurut catatan dari Korlantas Polri menyebutkan pengendara sepeda motor paling banyak melanggar aturan lalu lintas. sejak awal tahun 2023, Polri menindak 344 ribu sepeda motor yang melakukan pelanggaran. Salah satu pelanggaran yang dilakukan pengendara sepeda motor adalah tidak mengenakan helm. Padahal UU tentang LLAJ sudah menegaskan bahwa pengendara maupun penumpang yang tidak mengenakan helm berlogo Standar Nasional Indonesia (SNI) dapat dipidana kurungan paling lama satu bulan atau denda paling banyak 250 ribu rupiah (Pusiknas, 2023). Penggunaan helm Standar Nasional Indonesia (SNI) merupakan perlengkapan yang sangat diperlukan saat berkendara, karena helm tidak hanya dapat mengurangi risiko kecelakaan fatal, tetapi juga melindungi kepala dari dampak kerasnya benturan saat terjadi insiden di jalan.

Penerapan tilang di Indonesia sudah menggunakan sistem *Electronic Traffic Law Enforcement* (ETLE). ETLE adalah sistem penegakan hukum lalu lintas berbasis elektronik yang menggunakan teknologi canggih untuk mendeteksi, merekam, dan menindak pelanggaran lalu lintas secara otomatis. Sistem ini memanfaatkan perangkat seperti kamera CCTV, sensor, dan perangkat lunak pengenalan gambar untuk memantau lalu lintas di jalan raya dan mendeteksi pelanggaran seperti menerobos lampu merah, melanggar marka jalan, tidak menggunakan sabuk pengaman, menggunakan telepon genggam saat mengemudi, dan tidak memakai helm bagi pengendara sepeda motor.

Meskipun sistem ETLE menawarkan berbagai keunggulan dalam penegakan hukum lalu lintas, ETLE memiliki sejumlah kekurangan yang perlu diperhatikan. Salah

satu keterbatasannya adalah cakupan yang terbatas pada daerah dengan infrastruktur memadai, seperti kamera dan jaringan internet yang stabil, sementara daerah yang kurang berkembang mungkin belum memiliki fasilitas ini. Selain itu, sistem ETLE masih berfokus pada pelanggaran seperti tidak menggunakan sabuk pengaman, menerobos lampu merah, dan penggunaan pelat ganjil genap yang tidak sesuai dengan aturan (Kumparan, 2023).

Dalam melacak kendaraan sepeda motor yang tidak mengenakan helm, pihak berwajib perlu mengetahui pelat nomor dari kendaraan tersebut. Pelat nomor merupakan sistem penomoran unik untuk mengidentifikasi kendaraan bermotor yang diberlakukan di seluruh negara. Keberadaan pelat motor penting untuk menyatakan kendaraan milik seseorang sah di mata negara (Wihartasih & Wibawanto, 2015).

Ada berbagai penelitian yang mendeteksi helm pada pengendara sepeda motor, salah satunya adalah penelitian yang dilakukan oleh mahasiswa Universitas Telkom (Fuadi dkk., 2023), yang dimana penelitian ini menggunakan algoritma *Single Shot Detector* (SSD) yang menghasilkan sistem deteksi pelanggaran tidak menggunakan helm saat mengendarai sepeda motor. Oleh karena itu, pada penelitian ini akan membuat sebuah sistem deteksi dalam mencegah pelanggaran lalu lintas terhadap pengendara sepeda motor yang tidak memakai helm dengan menggunakan algoritma YOLO yang dimana sistem ini akan mendeteksi kendaraan, pelat, dan pengendara sepeda motor.

1.2 Teori

1.2.1 Helm dan Pelat Sepeda Motor

Helm merupakan bagian dari perlengkapan kendaraan bermotor berbentuk topi pelindung kepala yang berfungsi melindungi kepala pemakainya apabila terjadi benturan. Helm terbagi menjadi dua jenis yaitu *Open Face* (bentuk helm yang menutup kepala sampai dengan bagian leher menutup depan kuping) dan *Full Face* (bentuk helm yang menutup kepala atas, bagian leher, dan bagian mulut). Helm memiliki bagian keras dan halus, bagian luar dan bagian dalam dipasang untuk menyerap energi benturan, serta bagian muka helm yang dapat melindungi sebagian atau seluruh bagian muka terbuat dari lapisan bening (Susanto & Purnomo, 2022).

Dalam SNI 1811 – 2007 dicantumkan beberapa persyaratan mutu yang harus dipenuhi oleh suatu helm sebelum dapat digunakan antara lain (Pradypta, 2022):

1. Material helm dibuat dari bahan yang kuat dan bukan logam, tidak berubah jika ditempatkan di ruang terbuka pada suhu 0^oC – 50^oC selama paling tidak 4 jam dan tidak berpengaruh terhadap sinar ultraviolet, bensin, minyak, sabun, deterjen, air serta pembersih lainnya
2. Konstruksi helm harus memiliki tempurung yang keras dengan permukaan yang halus, lapisan peredam benturan, dan memiliki pengikat didagu. Tempurung helm terbuat dari bahan yang keras dengan ketebalan dan kemampuan homogen, serta tidak menyatu dengan pelindung muka.

Setiap kendaraan yang dioperasikan di jalan wajib memiliki pelat kendaraan atau sering juga disebut Tanda Nomor Kendaraan Bermotor (TNKB). Tanda Nomor Kendaraan Bermotor adalah suatu identitas kendaraan yang dikeluarkan oleh polisi dan

sering disebut juga sebagai nomor polisi. Setiap kendaraan memiliki kode nomor yang berbeda satu sama lainnya hal ini terjadi agar sewaktu-waktu tidak terjadi kecurangan oleh pengguna kendaraan dan mempermudah pihak yang berwenang (Halizah dkk., 2020).

Indonesia memiliki karakteristik pelat nomor tersendiri. Biasanya, format pada pelat nomor di Indonesia adalah **HH AAAA HH**, dimana **H** mempresentasikan huruf dan **A** mempresentasikan angka 0-9. Satu atau dua huruf pertama mempresentasikan area dimana kendaraan tersebut didaftarkan. Huruf ini diikuti dengan satu sampai empat angka, kemudian diakhiri oleh satu atau dua huruf. Misalnya, **BM 6015 TV** merupakan kendaraan dari Pekanbaru, karena huruf pertamanya, **BM**, merupakan kode untuk Pekanbaru. Sejak 2008, daerah Jakarta dan sekitarnya memperkenalkan format baru untuk pelat mobil, yaitu **HH AAAA HHH**. Sebelumnya, format ini telah diperkenalkan terlebih dahulu untuk pelat motor (Muchtamar, 2021).

1.2.2 Pengolahan Citra

Citra merupakan salah satu bentuk informasi yang diperlukan manusia selain teks, suara, dan video. Informasi ini diperlukan bukan hanya untuk komunikasi antar manusia saja tetapi juga antara manusia dan mesin. Inforamsi yang terkandung dalam sebuah citra dapat diinterpretasikan berbeda-beda oleh manusia satu dengan yang lain. Artinya, nilai informasi pada sebuah citra bersifat subjektif tergantung keperluan masing-masing manusia. Oleh karena itu, diperlukan pengolahan citra untuk mendapatkan citra yang memiliki informasi yang dikehendaki (Sulistiyanti dkk., 2016).

Berkembangnya teknik pengolahan citra dipicu oleh tujuan untuk membantu hidup manusia menjadi lebih mudah. Salah satu kemudahannya adalah membantu manusia menginterpretasikan objek yang tertangkap kamera menggunakan teknik peningkatan kualitas citra. Selain itu, pengolahan citra digunakan juga sebagai pengindra mesin otomatis. Perkembangan ini terbantu dengan perkembangan dunia teknologi khususnya komputer yang meningkat sangat cepat terutama segi kecepatan proses dan memori penyimpanan. Perkembangan ini memungkinkan pengolahan citra dapat dilakukan secara *real-time*, menyimpan citra dengan kapasitas memori yang lebih kecil tanpa mengurangi kualitas dan mengirimkan citra dengan cepat ke tempat yang berjarak jauh (Sulistiyanti dkk., 2016).

Dalam melakukan pengolahan citra, ada beberapa jenis citra digital yang sering digunakan dalam pengolahan citra, diantaranya, yaitu (Idris dkk., 2023):

1. **Citra Biner (Monokrom)**
   Citra monokrom yaitu gambar yang hanya terdiri dari dua warna, yaitu hitam (*black*) dan putih (*white*). Untuk menyimpan kedua warna ini dibutuhkan 1 bit pada memori. Nilai intensitas warna pada citra ini 0 (hitam) dan 1 (putih). Citra biner juga biasa dinamakan citra monokrom atau citra black and white. Proses pengolahan seperti segmentasi, pengembangan, morfologi, ataupun *dithering* seringkali menghasilkan citra biner. Citra biner banyak digunakan dalam pemrosesan citra, misalnya untuk kepentingan memperoleh tepi bentuk suatu objek.

2. Citra *Greyscale* (Skala Keabuan)

Citra *grayscale* merupakan citra yang derajat keabuannya memengaruhi nilai intensitas pikselnya. Pada citra *grayscale* atau monokrom, derajat warna hitam sampai dengan putih dibagi ke dalam 256 derajat keabuan di mana hitam sempurna dengan nilai 0 dan warna putih sempurna direpresentasikan dengan nilai 255.

Skala keabuan mengacu pada perubahan terkecil yang terlihat pada tingkat abu-abu. Mengukur perubahan tingkat keabuan adalah proses yang sangat subjektif dengan mengurangi bit R sambil memperbaiki resolusi spasial dan menciptakan kontur palsu. Hal ini disebabkan penggunaan tingkat abu-abu yang tidak mencukupi pada area halus gambar digital. Umumnya dapat terlihat pada gambar jarak keabuannya seragam. Tingkatan keabuan pada citra *grayscale* yaitu warna abu dengan berbagai level dari rentang hitam sampai mendekati putih seperti yang ditampilkan pada Gambar 1.

Gambar 1. Citra *Greyscale*

3. Citra Warna (*True Color*)

Kombinasi dari tiga warna dasar (RGB = *Red Green Blue*) terwakilkan oleh Setiap piksel pada citra warna. Digunakan penyimpanan 8 bit = 1 *byte* untuk setiap warna dasar, yang artinya memiliki gradasi sebanyak 255 warna untuk setiap warna. Terbentuk kombinasi warna sebanyak 16 juta warna lebih untuk setiap pikselnya. Karena mempunyai jumlah warna yang cukup besar sehingga bisa dikatakan hampir mencakup semua warna di alam yang menyebabkan format ini namakan *true color*.

Untuk citra *true color* RGB terdiri dari tiga kanal yaitu kanal merah, hijau dan biru. Setiap kanal warna memiliki nilai intensitas piksel dengan kedalaman bit 8 bit, yang berarti memiliki rentang warna 2^8 (0 hingga 255). Di saluran merah, 155 mewakili merah sempurna dan 0 mewakili hitam. Contoh citra *true color* dapat dilihat pada Gambar 2.

Gambar 2. Citra Warna

1.2.3 Video Digital

Video adalah teknologi untuk menangkap, merekam, memproses, mantransmisikan, dan menata ulang gambar bergerak. Biasanya menggunakan film seluloid, sinyal elektronik, atau media digital. Berkaitan dengan "penglihatan dan pendengaran" aplikasi video pada multimedia mencakup banyak aplikasi:

1. *Entertainment*: *roadcast* TV, VCR/DVD *recording*
2. *Interpersonal*: *video telephony*, *video conferencing*
3. *Interactive*: *windows*

Video digital adalah jenis sistem *video recording* yang bekerja menggunakan sistem digital dibandingkan dengan analog dalam hal representasi videonya. Biasanya digital video direkam dalam *tape*, kemudian didistribusikan melalui *optical disc*, misalnya VCD dan DVD. Salah satu alat yang dapat digunakan untuk menghasilkan video digital adalah *camcorder*, yang digunakan untuk merekam gambar-gambar video dan audio, sehingga *camcorder* akan terdiri dari *camera* dan *recorder* (Sadono, 2012).

Jenis file video digital sangat beragam sehingga tidak semua aplikasi pemutar video (*video player*) mampu memutar semua jenis *file* video. Adapun di antara jenis *file* video diantaranya, yaitu (Batubara & Ariani, 2016):

AVI (*Audio Video Interleaved*) adalah format *file* video buatan Microsoft yang tidak dikompresi sehingga ukuran filenya cukup besar dan memiliki gambar yang tajam. *File* video jenis ini dapat diputar pada komputer menggunakan *windows media player*.

MPEG (*Moving Picture Experts Group*) adalah standar kompresi *file* digital video-audio untuk disimpan dalam sebuah media penyimpanan seperti CD dan VCD.

WMV (*Windows Media Video*) adalah format standar *windows* yang tidak banyak digunakan sebagai standar *video editing*. WMV merupakan gabungan dari AVI dan WMA yang terkompres dan berektensi ".wmv".

DiVX adalah salah satu codec video yang diciptakan oleh DiVXNewtworks. Format video ini menggunakan kompresi berbasi MPEG-4 sehingga memiliki ukuran yang sangat kecil, bahkan dapat mencapai kurang dari seperdelapan ukuran MPEG-2 dengan kualitas tetap terjaga.

MP4 adalah format video yang banyak disimpan di internet. Sebagian pemutar video belum dapat memutar format file video ini. Oleh karena itu, pengguna perlu memasang aplikasi pemutar video khusus untuk bisa memainkan format video ini.
MOV adalah format video terkompresi yang dibuat oleh *Apple* komputer dan dijalankan pada *platform* Macintosh, dan bisa juga dijalankan pada *windows* dengan memasang aplikasi *Quick Time*. Format ini ditujukan untuk *online video* dan *website* yang berbasis multimedia.
RealMedia adalah format video yang dirancang untuk keperluan *streaming* dan dapat menampung *file* berupa video, audio, animasi, MIDI, serta presentasi. Transmisinya menggunakan *Rapid Spanning Tree Protocol* (RSTP). Codec yang biasanya ada di dalam *file* video *RealMedia* adalah *RealVideo*.
ASF (*Advance Streaming Format*) dikeluarkan oleh Microsoft untuk keperluan *streaming*. Ada tiga bagian yang terkandung dalam *file* ASF: *Header* terkandung dalam *file* ASF, objek data termasuk media *streaming*, dan objek indeks opsional yang memberikan kontribusi untuk mengaktifkan akses acak ke data dalam *file*.
3GP (3GPP *Format File*) adalah format video hasil rekaman perangkat komunikasi *mobile* (*handphone*). Format 3GP menggunakan kecepatan putar 15 *frame* per detik. Dalam praktek, format video 3GPP dapat dimainkan oleh *QuickTime Player* 7 dan *Windows Media Player*.
FLV (*Flash Video*) adalah format video yang biasa digunakan untuk menyisipkan video ke dalam halaman web, seperti *Youtube*, *Facebook*, dan lain sebagainya.

1.2.4 Deteksi Objek

Deteksi objek melibatkan penemuan instansi objek dari satu atau beberapa kelas dalam sebuah gambar. Tujuan dari deteksi objek adalah untuk mendeteksi semua instansi objek dari satu atau beberapa *class* yang diketahui, seperti manusia, mobil, atau wajah dalam sebuah gambar. Biasanya, hanya sedikit objek yang ada dalam gambar, tetapi terdapat jumlah lokasi dan skala yang sangat besar di mana objek tersebut dapat muncul dan perlu dijelajahi dengan cara tertentu.

Sistem deteksi objek membangun model untuk kelas objek dari sejumlah contoh *training*. Dalam kasus objek yang kaku dan tetap, mungkin hanya diperlukan satu contoh, tetapi umumnya diperlukan beberapa contoh *training* (sering ratusan atau ribuan) untuk menangkap aspek-aspek tertentu dari variasi *class*. Secara umum, diperlukan lebih sedikit data *training* ketika lebih banyak informasi tentang variasi *class* dapat secara eksplisit dimasukkan ke dalam model. Namun, mungkin sulit untuk menentukan model yang menangkap variasi yang luas yang ditemukan dalam gambar. Salah satu contohnya adalah menggunakan metode seperti *Convolutional Neural Network* (CNN) yang mempelajari variasi *class* dari *big data* (Amit dkk., 2020).

Masalah dari deteksi objek adalah menentukan menentukan di mana objek berada dalam suatu gambar yang diberikan (*object localization*) dan harus menentukan kategori setiap objek dari gambar tersebut (*object classification*). Oleh karena itu, secara garis besar model deteksi objek dapat dibagi menjadi tiga tahap utama (Zhao dkk., 2019):

Pemilihan wilayah yang informatif: Karena berbagai objek dapat muncul di posisi apa pun dalam gambar dan memiliki rasio aspek atau ukuran yang berbeda,

pemindaian seluruh gambar dengan jendela geser berbagai skala merupakan pilihan yang tepat.

***Feature Extraction*:** Untuk mengenali berbagai objek, diperlukan mengekstrak fitur visual yang dapat memberikan representasi semantik dan tangguh. Hal ini disebabkan oleh kenyataan bahwa fitur-fitur ini dapat menghasilkan representasi yang terkait dengan sel-sel kompleks dalam otak manusia. Namun, karena keragaman penampilan, kondisi pencahayaan, dan *background*, sulit untuk merancang secara manual *feature descriptor* yang tangguh untuk secara sempurna menggambarkan semua jenis objek.

***Classification*:** Diperlukan suatu pengklasifikasi untuk membedakan objek target dari semua kategori lainnya dan membuat representasi menjadi lebih hierarkis, semantik, dan informatif untuk pengenalan visual. Biasanya, *Support Vector Machine* (SVM), *AdaBoost*, dan *Deformable Part-based Model* (DPM) adalah pilihan model yang terbaik. Di antara pengklasifikasi ini, DPM adalah model yang fleksibel dengan menggabungkan bagian objek dengan biaya deformasi untuk menangani deformasi yang parah. Dalam DPM, dengan bantuan model grafis, fitur tingkat rendah yang dirancang dengan hati-hati dan dekomposisi bagian yang terinspirasi secara kinematika digabungkan.

1.2.5 Visi Komputer

Visi komputer adalah ilmu dan teknologi mesin yang memiliki fitur untuk melihat, di mana mesin mampu mengekstrak informasi dari gambar yang diperlukan untuk menyelesaikan tugas tertentu. Sebagai suatu disiplin ilmu, visi komputer berkaitan dengan teori dalam bagian sistem buatan mengenai ekstrak informasi dari gambar. Data gambar dapat mengambil banyak bentuk, seperti urutan video, pandangan dari beberapa kamera atau data multi-dimensi dari *scanner* medis. Sedangkan sebagai disiplin teknologi, visi komputer berusaha untuk menerapkan teori dan model untuk pembangunan sistem visi komputer.

Visi komputer didefinisikan sebagai salah satu cabang ilmu pengetahuan yang mempelajari bagaimana komputer dapat mengenali objek yang diamati. Cabang ilmu ini bersama *artificial intelligence* akan mampu menghasilkan *visual intelligence system*. Pendapat lain mengatakan visi komputer adalah proses otomatis yang mengintegrasikan sejumlah besar proses untuk persepsi visual, seperti akusisi citra, pengolahan citra, pengenalan dan pembuatan keputusan. Visi Komputer mencoba meniru cara kerja sistem visual manusia (*human vision*) yang sesungguhnya sangat kompleks, bagaimana manusia melihat objek dengan indra penglihatan (mata), lalu citra objek diteruskan ke otak untuk diinterpretasi sehingga manusia mengerti objek apa yang tampak dalam pandangan mata. Selanjutnya hasil interpretasi ini digunakan untuk pengambilan keputusan (Amrizal & Aini, 2013).

Penerapan visi komputer pada kehidupan manusia beragam pada berbagai aspek. Hal ini dikarenakan efektivitas bantuan mesin dalam menganalisis citra baik 2D dan 3D terbukti dan terjamin hasilnya. Berbagai aspek krusial yang membutuhkan pembuatan keputusan penting bahkan sudah menerapkan bantuan alat pendeteksi citra,

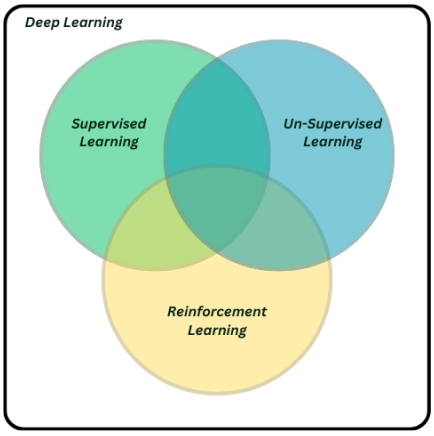
seperti contohnya dalam bidang kesehatan dengan adanya klasifikasi sel tumor dari hasil rontgen (Arnita dkk., 2022).

1.2.6 *Deep Learning*

Deep Learning (DL) adalah salah satu cabang dari *Machine Learning* yang terdiri dari algoritma pemodelan abstraksi tingkat tinggi pada data menggunakan sekumpulan fungsi transformasi non-linear yang ditata berlapis-lapis dan mendalam. *Deep learning* sangat baik untuk diterapkan pada *supervised learning*, *unsupervised learning*, dan *semi-supervised learning* maupun untuk *reinforcement learning* dalam berbagai aplikasi seperti pengenalan citra, suara, klasifikasi teks, dan sebagainya (Cholissodin & Soebroto, 2019).

Deep learning, yang mulai populer digunakan sejak tahun 2006, menggunakan mekanisme *deep architecture of learning* atau pendekatan *hierarchical learning*. *Learning* atau pembelajaran dalam hal ini adalah sebuah prosedur yang berisi proses estimasi parameter-parameter suatu model sehingga model yang dikembangkan (algoritma) dapat menyelesaikan suatu tugas atau permasalahan tertentu.

Deep learning menggunakan beberapa lapisan (*layers*) di antara lapisan masukan (*input layer*) dan lapisan keluaran (*output layer*). Arsitektur tersebut dapat digunakan untuk melakukan pemrosesan nonlinier dengan beberapa tahap yang hasilnya dapat digunakan untuk *feature learning* dan klasifikasi pola (*pattern classification*). Jumlah lapisan dalam *deep learning* yang bervariasi dapat digunakan untuk melakukan abstraksi dengan tingkat yang berbeda-beda.



Gambar 3. Kategori dalam *Deep Learning* (DL)

Beberapa teknik dalam *deep learning* dapat dikategorikan menjadi *supervised*, *semi-supervised*, dan *unsupervised*. Kategori lain, seperti *Reinforcement Learning* (RL) atau *Deep Reinforcement Learning* (DRL), seringkali dikategorikan menjadi *semi-supervised* atau *unsupervised*. Pengategorian tersebut diperlihatkan pada Gambar 3 (Diponegoro dkk., 2021):

***Deep Supervised Learning*:** Teknik *learning* yang digunakan dalam kategori ini menggunakan data yang telah diberi label sebelumnya (*labeled data*). Contoh yang populer dalam kategori ini adalah *Deep Neural Networks* (DNN),

Convolutional Neural Network (CNN), *Recurrent Neural Network* (RNN), termasuk juga *Long Short-Term Memory* (LSTM), dan *Gated Recurrent Unit* (GRU).

***Deep Semi-Supervised Learning*:** *Semi-supervised learning* menggunakan teknik *learning* yang menggunakan sebagian data yang telah diberi label sebelumnya (*partially labeled data*). Pada beberapa kasus, DRL, *Generative Adversarial Networks* (GAN), serta RNN, termasuk LSTM, dan GRU juga menggunakan teknik *learning* ini.

***Deep Unsupervised Learning*:** Teknik *learning* ini menggunakan data yang tidak diberi label sebelumnya (*unlabeled data*). *Auto Encoders* (AE), *Restricted Boltzmann Machines* (RBM), dan generasi terbaru dari GAN menggunakan teknik *learning* ini pada implementasinya.

***Deep Reinforcement Learning*:** Teknik *learning* ini digunakan pada lingkungan atau *environments* yang tidak diketahui (*unknown environments*). Pada tahun 2013, DRL dimulai dengan hadirnya *Google DeepMind*.

1.2.7 Python & OpenCV

Python adalah bahasa pemrograman interpretatif multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. Python diklaim sebagai bahasa yang menggabungkan kapabilitas, kemampuan, dengan sintaksis kode yang sangat jelas, dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif. Python juga didukung oleh komunitas yang besar. Python mendukung multi paradigma pemrograman, utamanya, namun tidak dibatasi pada pemrograman berorientasi objek, pemrograman imperatif, dan pemrograman fungsional. Salah satu fitur yang tersedia pada python adalah sebagai bahasa pemrograman dinamis yang dilengkapi dengan manajemen memori otomatis. Seperti halnya pada bahasa pemrograman dinamis lainnya, python umumnya digunakan sebagai bahasa skrip meski pada praktiknya penggunaan bahasa ini lebih luas mencakup konteks pemanfaatan yang umumnya tidak dilakukan dengan menggunakan bahasa skrip. Python dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan di berbagai platform sistem operasi.

Python didistribusikan dengan beberapa lisensi yang berbeda dari beberapa versi. Namun pada prinsipnya Python dapat diperoleh dan dipergunakan secara bebas, bahkan untuk kepentingan komersial. Lisensi Python tidak bertentangan baik menurut definisi *Open Source* maupun *General Public License* (GPL). Python dikembangkan oleh Guido van Rossum pada tahun 1990 di Stichting Mathematisch Centrum (CWI), Amsterdam sebagai kelanjutan dari bahasa pemrograman ABC. Saat ini pengembangan Python terus dilakukan oleh sekumpulan pemrogram yang dikoordinir Guido dan *Python Software Foundation*. *Python Software Foundation* adalah sebuah organisasi non-profit yang dibentuk sebagai pemegang hak cipta intelektual Python sejak versi 2.1 dan dengan demikian mencegah Python dimiliki oleh perusahaan komersial. Saat ini distribusi Python sudah mencapai versi 2.7.4 dan versi 3.6.3. Nama python dipilih oleh Guido sebagai nama bahasa ciptaannya karena kecintaan Guido pada acara televisi *Monty Python's Flying Circus*. Oleh karena itu seringkali ungkapan-ungkapan khas dari

acara tersebut seringkali muncul dalam korespondensi antar pengguna python (Buana, 2018).

OpenCV (*Open Computer Vision*) merupakan sebuah API (*Application Programming Interface*) *library open source* yang sangat cocok digunakan untuk *image processing*. OpenCV pertama kali dibuat oleh Intel pada tahun 1999 oleh Gary Bradsky dan mulai dirilis keluar pada tahun 2000. Pada tahun 2005, OpenCV berhasil memenangkan DARPA *Grand Challenge*. Saat ini, OpenCV telah mendukung banyak algoritma yang terkait dengan *Computer Vision* (CV) dan *Machine Learning* (ML). Selain itu, saat ini OpenCV juga dapat digunakan dalam berbagai macam bahasa pemrograman, seperti C++, Python, Java, dan lain sebagainya. Tidak hanya itu, OpenCV juga tersedia dalam berbagai platform, seperti *Windows*, Linux, OSX, Android, IOS, dan lain sebagainya. OpenCV juga dapat melakukan operasi *high-speed* GPU dengan menggunakan antarmuka-antarmuka berbasis CUDA dan OpenCL. Kombinasi terbaik untuk dapat melakukan operasi berkecepatan tinggi tersebut adalah dengan perpaduan antara OpenCV, C++ API dan bahasa pemrograman Python.

OpenCV memiliki banyak sekali fitur yang dapat dimanfaatkan, berikut adalah beberapa fitur utama dari OpenCV (Gumelar, 2019):

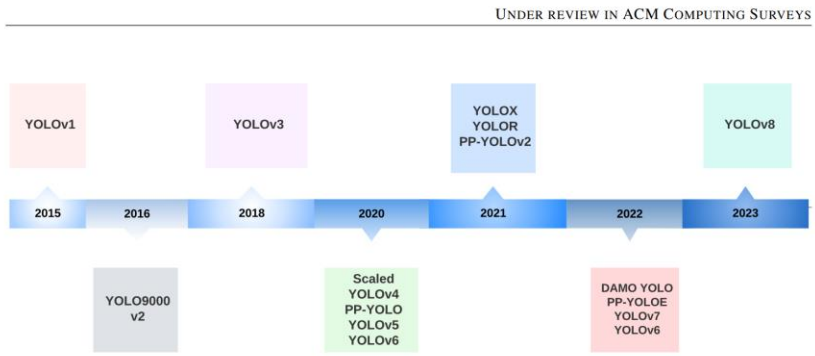
1. ***Image* dan *Video* I/O**
   Fitur ini memungkinkan untuk membaca data gambar dari *file* atau dari umpan video langsung. Begitu pula sebaliknya dapat menciptakan *file* gambar maupun video.
2. ***Computer Vision* dan *Image Processing* (API tingkat *low* dan *mid*)**
   Interface ini dapat melakukan eksperimen uji coba dengan berbagai standar algoritma *computer vision*, seperti *line detection*, *edges detection*, proyeksi elisps, *image pyramid*, *template matching*, berbagai macam transformasi (Fourier, *discrete cosine*, *distance transform*), dan lain sebagainya.
3. ***Computer Vision High Level***
   OpenCV juga memungkinkan penggunanya untuk melakukan operasi tingkat tinggi, seperti mendeteksi wajah, pengenalan wajah, *optical flow*, dan lain sebagainya.
4. **AI (*Artificial Intelegent*) dan ML (*Machine Learning*)**
   OpenCV juga menyediakan paket library ML (*Machine Learning*), tentunya hal ini akan sangat mendukung komputer dalam mendeteksi, mengambil keputusan, dan melakukan aksi terhadap suatu objek.
5. **Sampling Gambar dan Substraksi**
   Fitur ini meliputi substraksi *subregion* dari gambar, *random sampling*, *rotating*, dan lain sebagainya.
6. **Gambar Biner**
   OpenCV juga memiliki fitur untuk dapat membuat dan menganalisis gambar biner.
7. **Pemodelan 3D**
   Fitur ini sangat bermanfaat untuk *mapping* dan *localization*, baik menggunakan *stereo* kamera maupun *single* kamera dengan berbagai sudut pandang.

1.2.8 *You Only Look Once* Versi 8 (YOLOv8)

You Only Look Once (YOLO) adalah sebuah algoritma yang dikembangkan untuk mendeteksi sebuah objek secara *real-time*. Sistem pendeteksian yang dilakukan adalah dengan menggunakan *repurpose classifier* atau *localizer* untuk melakukan deteksi. sebuah model diterapkan pada sebuah citra di beberapa lokasi dan skala daerah dengan citra yang diberi *score* paling tinggi akan dianggap sebagai sebuah pendeteksian.

YOLO menggunakan pendekatan Jaringan Saraf Tiruan (JST) untuk mendeteksi objek pada sebuah citra. Jaringan ini membagi citra menjadi beberapa wilayah dan memprediksi setiap kotak pembatas dan probabilitas untuk setiap wilayah. Kotak-kotak pembatas ini kemudian dibandingkan dengan probabilitas yang diprediksi. YOLO memiliki beberapa kelebihan dibandingkan dengan sistem yang berorientasi pada *classifier*, terlihat dari seluruh citra pada saat dilakukan test dengan prediksi yang diinformasikan secara global pada citra (Rahma dkk., 2021).

Deteksi objek secara *real-time* telah menjadi komponen penting dalam banyak aplikasi, melibatkan berbagai bidang seperti kendaraan otonom, robotika, pemantauan video, dan realitas tercambah. Di antara berbagai algoritma deteksi objek, kerangka kerja YOLO (*You Only Look Once*) telah menonjol karena keseimbangan yang luar biasa antara kecepatan dan akurasi, memungkinkan identifikasi objek yang cepat dan andal dalam gambar. Sejak awal, YOLO telah berkembang melalui beberapa iterasi, setiap versi membangun pada versi sebelumnya untuk mengatasi keterbatasan dan meningkatkan kinerja (lihat Gambar 4) (Terven & Cordova-Esparza, 2023).



Gambar 4. *Timeline* Versi Model YOLO

Untuk mendeteksi suatu objek, YOLO menggunakan model yang terpadu di mana jaringan konvolusi tunggal secara simultan memprediksi beberapa kotak pembatas (bounding boxes) dan probabilitas kelas dalam kotak-kotak tersebut. Berikut adalah beberapa langkah You Only Look Once (YOLO) untuk deteksi objek (Rezkiani dkk., 2022):

1. Algoritma YOLO akan membagi gambar yang dimasukkan menjadi *grid* S x S dengan membuat kotak pembatas untuk setiap objek dengan menetapkan skor kepercayaan (*confidence score*) dan label kelas. *Confidence score* mengindikasikan seberapa akurat model dalam mendeteksi bahwa ada objek

dalam kotak tersebut. Teknik klasifikasi gambar dan lokalitas objek diterapkan pada setiap *grid* gambar dan memberikan label pada setiap *grid*. Nilai *confidence score* diperoleh menggunakan persamaan (1).

$$Confidence\ score = \Pr(Class) \times IoU_{pred}^{truth} \quad (1)$$

Intersection over Union (IoU) dapat dihitung dengan membandingkan kotak pembatas ground-truth dan bounding box yang diprediksi, seperti pada persamaan (2).

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} \quad (2)$$

2. Selanjutnya memerikasa setiap *grid* secara terpisah dan menandai label yang melewati objek di dalamnya beserta *bounding box* atau kotak pembatasnya. Label *grid* tanpa objek ditandai sebagai 0. Setiap sel *grid* memprediksi (x, y, w, h) dan *confidence score* untuk setiap objek. *Confidence score* mengukur akurasi objek dalam *bounding box*. Semua nilai (x, y, w, h) bearada antara 0 dan 1.

YOLOv8 adalah model YOLO yang dirilis pada tahun 2023 yang dapat digunakan untuk deteksi objek, klasifikasi gambar, dan tugas segmentasi secara instan. YOLOv8 dikembangkan oleh Ultralytics, yang juga menciptakan model YOLOv5 yang berpengaruh dan mendefinisikan industri. YOLOv8 mencakup banyak perubahan dan perbaikan arsitektural dan pengembangan dari YOLOv5.

YOLOv8 sedang dalam pengembangan aktif pada masa sekarang ini, saat Ultralytics bekerja pada fitur-fitur baru dan merespons umpan balik dari komunitas. Sebenarnya, ketika Ultralytics merilis suatu model, model tersebut mendapatkan dukungan jangka panjang. Organisasi tersebut bekerja sama dengan komunitas untuk membuat model tersebut semaksimal mungkin. YOLOv8 mencapai akurasi yang tinggi pada *dataset* COCO. Sebagai contoh, model YOLOv8m mencapai *mean average precision* (mAP) sebesar 50,2% saat diukur pada COCO. Ketika dievaluasi terhadap Roboflow, sebuah *dataset* yang secara khusus mengevaluasi kinerja model pada berbagai domain tugas khusus, YOLOv8 mencetak skor yang jauh lebih baik dibandingkan YOLOv5.

Selain itu, fitur kenyamanan bagi pengembang dalam YOLOv8 sangat signifikan. Berbeda dengan model lain di mana tugas-tugas terbagi di banyak *file* Python yang dapat dijalankan, YOLOv8 dilengkapi dengan *Command Line Interface* (CLI) yang membuat pelatihan model lebih intuitif. Ditambah dengan paket Python yang memberikan pengalaman penulisan kode yang lebih mulus dibandingkan dengan model sebelumnya (Telaumbanua dkk., 2023). Hal ini dikarenakan arsitektur dari YOLOv8 yang didesain agar cepat, akurat, dan lebih mudah digunakan.

1.2.9 *Confusion Matrix*

Confusion matrix merupakan tabel pencatat hasil kerja klasifikasi. Matriks Konfusi berisi informasi tentang klasifikasi yang dapat diprediksi dengan sistem klasifikasi. Kinerja sistem umumnya dievaluasi menggunakan data dalam bentuk matriks (Santra & Christy,

2012). Melalui keakuratan *confusion matrix*, tingkat kesalahan, ketepatan, dan nilai penarikan dapat diketahui. Kuantitas *confusion matrix* dapat diringkus menjadi 2 nilai, yaitu akurasi dan laju *error*. Dengan mengetahui jumlah data yang diklasifikasikan dengan benar, diketahui pula akurasi hasil prediksi dan dengan mengetahui jumlah data yang diklasifikasikan secara salah, diketahui pula laju *error* dari prediksi yang dilakukan.

Pada pengukuran kinerja menggunakan *confusion matrix*, terdapat 4 istilah sebagai representasi hasil proses klasifikasi. Keempat istilah tersebut adalah *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). Nilai *True Negative* merupakan jumlah data negatif yang terdeteksi dengan benar, sedangkan *False Positive* merupakan data negatif namun terdeteksi sebagai data positif. Sementara itu, *True Positive* merupakan data positif yang terdeteksi benar. *False Negative* merupakan kebalikan dari *True Positive*, sehingga data positif, namun terdeteksi sebagai data negatif.

Berdasarkan nilai *True Negative*, *False Positive*, *False Negative*, dan *True Positive* dapat diperoleh nilai akurasi, presisi, *error*, dan *recall*. Namun untuk penelitian ini hanya akan membahas tentang akurasi dan *error* pada sistem. Akurasi menggambarkan seberapa akurat sistem dapat mengklasifikasi data secara benar. Dengan kata lain, nilai akurasi merupakan perbandingan antara data yang terklasifikasi benar dengan keseluruhan data. Sedangkan *error* merupakan seberapa tidak akuratnya sistem dalam mengklasifikasikan data secara benar. Dengan kata lain, nilai akurasi merupakan perbandingan antara data yang terklasifikasi benar dengan keseluruhan data. Nilai akurasi dapat diperoleh dengan persamaan (3).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \times 100\% \quad (3)$$

Selain itu, untuk mencari akurasi dari sebuah *confusion matrix* pada kasus *multi-class*, dapat menggunakan persamaan (4).

$$Accuracy = \frac{\sum_{i=1}^{k}(TP_i+TN_i)}{N} \times 100\% \quad (4)$$

Nilai *error rate* atau *misclassification rate* dapat ditentukan dengan menunjukkan seberapa sering model melakukan kesalahan dalam prediksi. Untuk mengetahui nilai *error rate* pada *confusion matrix* dapat dilihat pada persamaan (5).

$$Error\ Rate = \frac{FP+FN}{TP+TN+FP+FN} \times 100\% \quad (5)$$

Selain itu, untuk mencari *error rate* dari sebuah *confusion matrix* pada kasus *multi-class*, dapat menggunakan persamaan (6).

$$Error\ Rate = \frac{\sum_{i=1}^{k}(FP_i+FN_i)}{N} \times 100\% \quad (6)$$

Adapun rumus untuk mendapatkan nilai *error rate* dapat dinyatakan juga pada persamaan (7).

$$Error\ Rate = 100\% - Accuracy \quad (7)$$

Selanjutnya, jika ingin mengetahui perbandingan nilai antara *True Positive* dengan banyaknya data yang diprediksi positif, dapat dilihat pada persamaan (8).

$$Precision = \frac{TP}{TP+FP} \times 100\% \quad (8)$$

Jika ingin mengetahui perbandingan nilai antara *True Positive* dengan banyaknya data yang sebenarnya positif, dapat dilihat pada persamaan (9).

$$Recall = \frac{TP}{TP+FN} \times 100\% \quad (9)$$

Diketahui, jika nilai salah satu dari *precision* atau *recall* tinggi, maka nilai salah satu yang lain cenderung lebih rendah. Sebagai contoh jika nilai *precision* sangat tinggi atau mendekati skor 1, maka nilai dari persamaan *recall* akan cenderung lebih rendah. Jika hanya menggunakan 2 persamaan tersebut, dapat terjadi bias pada kondisi dunia nyata. Oleh sebab itu tidak boleh hanya menggunakan skor tersebut untuk melakukan pemilihan model. Untuk mengatasi hal ini, diperlukan persamaan *F1-Score* yang dituliskan sebagai persamaan (10).

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \times 100\% \quad (10)$$

Nilai yang didapatkan kemudian menggunakan rumus *F1-Score* berfungsi sebagai *harmonic mean* dari *precision* dan *recall*. Nilai terbaik dari *F1-Score* adalah 1 dan nilai terburuk adalah 0. Secara representasi, jika *F1-Score* punya skor yang baik mengindikasikan bahwa model klasifikasi memiliki *precision* dan *recall* yang baik pula (Anwar, 2022).

1.2.10 Roboflow

Roboflow adalah platform yang diciptakan untuk mempermudah proses pengembangan dan pelatihan model dalam bidang *computer vision*, yang menggunakan data berupa gambar atau video. Platform ini memberikan fasilitas kepada pengembang dan ilmuwan data untuk secara efisien melakukan pemrosesan, manajemen, dan penandaan (*labeling*) terhadap data pelatihan, serta melatih model *computer vision* yang kuat, tanpa harus menghadapi kendala teknis yang rumit.

Dalam penggunaannya, Roboflow berfungsi sebagai alat bantu yang bertujuan untuk memfasilitasi tahapan-tahapan yang diperlukan dalam proses pengembangan model *computer vision*. Dalam platform ini, pengguna dapat mengelola dan mengatur *dataset* gambar atau video yang akan digunakan sebagai data pelatihan untuk model. Mengunggah data, menentukan label, dan mengelola anotasi untuk mengidentifikasi objek atau fitur yang ada dalam setiap gambar.

Salah satu fitur utama yang ditawarkan oleh Roboflow adalah kemampuannya dalam mengotomatisasi proses pelabelan data. Proses ini biasanya memakan waktu dan

bisa menjadi pekerjaan yang berulang terutama ketika dataset sangat besar. Roboflow menggunakan teknik pemrosesan citra dan deep learning untuk membantu dalam tugas ini, sehingga dapat menghemat waktu dan usaha yang diperlukan untuk melakukan pelabelan secara manual.

Setelah data telah diberi label, Roboflow juga menyediakan fasilitas untuk membagi *dataset* menjadi beberapa bagian, seperti pelatihan (*training*), validasi (*validation*), dan pengujian (*testing*), yang merupakan praktik umum dalam pelatihan model *deep learning*. Di samping itu, platform ini mendukung berbagai jenis model *computer vision* yang populer, seperti YOLO (*You Only Look Once*), *Faster* R-CNN, dan lainnya.

Selanjutnya, Roboflow memudahkan pengguna dalam melatih model dengan menggunakan *dataset* yang telah siap. Mengatur parameter pelatihan, melakukan optimisasi kinerja model, dan memantau metrik evaluasi seperti akurasi dan *loss*. Roboflow merampingkan proses ini sehingga pengembang bahkan yang belum memiliki pengetahuan mendalam dalam *deep learning* tetap dapat membangun dan melatih model *computer vision* yang efektif.

Secara keseluruhan, Roboflow merupakan solusi yang membantu pengembang dan ilmuwan data dalam mengatasi beberapa kendala teknis dalam mengembangkan model *computer vision*. Platform ini mempercepat proses pengembangan model dengan mengatasi hambatan-hambatan teknis yang mungkin terjadi, seperti pelabelan data yang kompleks, pemilihan model yang tepat, dan pelatihan model yang efisien (Putra, 2023).

1.3 Rumusan Masalah

1. Bagaimana cara sistem mampu mendeteksi helm dan pelat pada pengendara sepeda motor dalam mencegah pelanggaran lalu lintas dengan menggunakan algoritma YOLO?
2. Bagaimana kinerja sistem deteksi yang telah dibuat dalam mendeteksi helm dan pelat pada pengendara sepeda motor dengan menggunakan algoritma YOLO?

1.4 Tujuan Penelitian

1. Membuat sistem deteksi yang mampu mendeteksi helm dan pelat pada pengendara sepeda motor dalam mencegah pelanggaran lalu lintas dengan menggunakan algoritma YOLO.
2. Mengetahui kinerja dari sistem deteksi yang dibuat untuk mendeteksi helm dan pelat pada pengendara sepeda motor dengan menggunakan algoritma YOLO.

1.5 Manfaat Penelitian

1. Penelitian ini bermanfaat untuk membantu sistem keamanan pihak kepolisian dalam memantau pengendara yang tidak mengenakan helm saat mengendarai sepeda motor.
2. Penelitian ini bermanfaat sebagai referensi bagi peneliti dalam pengembangan topik terkait sistem deteksi helm dan pelat pada pengendara sepeda motor pada bidang visi komputer.

1.6 Ruang Lingkup

1. Pengambilan data dilakukan di siang hari.
2. Pengambilan data dilakukan secara langsung menggunakan kamera *handphone* dengan spesifikasi kamera 12 MP dan frame rate 30 fps.
3. Data yang diolah berupa data video dengan resolusi 1920x1080.
4. Objek yang dideteksi adalah pengendara, pelat, dan kendaraan sepeda motor.
5. Jarak kamera dengan objek yang dideteksi adalah sekitar 13 meter.

BAB II

METODE PENELITIAN

2.1 Waktu dan Lokasi Penelitian

Penelitian ini dimulai pada tanggal 31 Mei 2023. Untuk lokasi menguji, menganalisis serta penulisan program dilakukan di rumah dan di Laboratorium Animasi dan Multimedia Teknik Informatika Universitas Hasanuddin. Adapun lokasi pengambilan data video untuk *training data* dan *testing data* dilakukan di Jembatan Penyebrangan Urip yang beralamat Jalan Urip Sumoharjo. Adapun lokasi pengambilan data dapat dilihat pada Gambar 5.

Gambar 5. Lokasi Pengambilan Data

2.2 Instrumen Penelitian

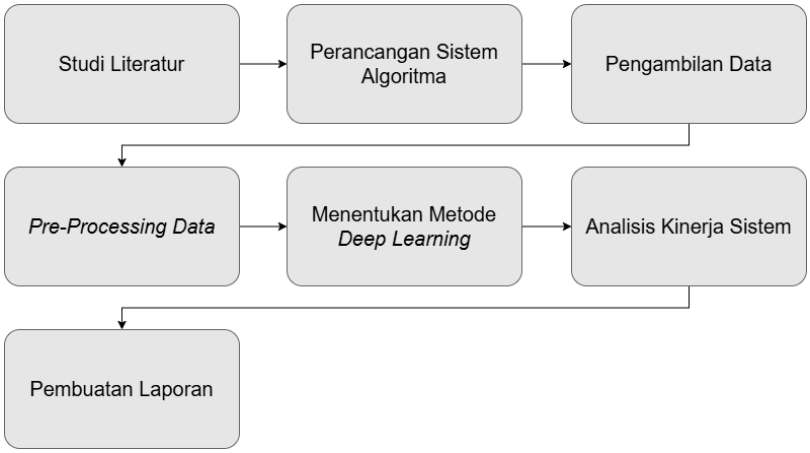
Pada penelitian kali ini, peneliti menggunakan dua jenis komponen instrumen penelitian yaitu perangkat lunak (*software*) dan perangkat keras (*hardware*). Berikut ini adalah beberapa instrumen yang digunakan pada masing-masing komponen:

1. **Perangkat Lunak (*Software*)**
   Berikut ini adalah beberapa perangkat lunak (*software*) yang digunakan pada penelitian ini:
   a. PyCharm 2023.2.5
   b. Google Colaboratory
   c. Windows 11 64 bit
   d. Roboflow
   e. Python 3.10.1
   f. OpenCV 4.9.0
   g. Streamlit 1.30.0
   h. YOLOv8
2. **Perangkat Keras (*Hardware*)**
   Berikut ini adalah beberapa perangkat keras (*hardware*) yang digunakan pada penelitian ini:

a. ASUS VivoBook 14/15 Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz
b. Kamera *handphone* OPPO A5 2020 dengan spesifikasi kamera 12 MP
c. Tripod Kamera 135 cm

2.3 Tahapan Penelitian

Berikut ini adalah beberapa tahapan yang akan dilakukan pada penelitian ini dapat dilihat pada Gambar 6.



Gambar 6. Tahapan Penelitian

1. **Studi Literatur**
   Pada tahap ini, dilakukan pencarian penelitian terkait dari berbagai sumber yang berhubungan dengan penelitian tentang sistem helm dan pelat nomor kendaraan sepeda motor. Sistem ini juga akan mendeteksi kendaraan sepeda motor. Penelusuran literatur ini dilakukan untuk mendapatkan informasi terbaru dan mendalam mengenai kemajuan, teknik, dan hasil penelitian sebelumnya dalam topik ini.
2. **Menentukan Metode *Deep Learning***
   Pada tahap ini, peneliti akan memilih metode *deep learning* yang akan digunakan berdasarkan hasil temuan dari penelitian-penelitian sebelumnya agar sistem pada penelitian ini dapat berjalan dengan baik. Model *deep learning* yang akan digunakan pada penelitian ini adalah algoritma YOLOv8.
3. **Pengambilan Data**
   Dalam Tahap ini, peneliti akan mengumpulkan data video dengan menggunakan kamera *handphone* untuk mengambil data secara langsung dari jalan raya dengan *Point of View* (POV) dari atas penyeberangan Jalan Urip Sumoharjo. Data video yang diambil akan berperan sebagai *dataset* pelatihan dan pengujian untuk sistem e-tilang yang akan dikembangkan.

4. ***Preprocessing Data***
   Pada tahap ini, dilakukan serangkaian langkah *cleaning data* pada *dataset* yang telah terhimpun. *Preprocessing* ini bertujuan untuk menganalisis *dataset* yang telah dikumpulkan guna mengidentifikasi dan mengeliminasi data yang tidak relevan atau tidak dapat digunakan dalam pengembangan sistem e-tilang yang akan dibuat.
   Pada tahap ini, *preprocessing* yang dipilih adalah *resize* dan augmentasi data. Proses *resize* mampu mengubah dimensi tinggi dan lebar gambar dalam *dataset* tanpa mengubah aspek rasio atau konten visualnya. Jadi, ketika proses pengambilan *dataset* telah dilakukan, selanjutnya yaitu mengubah resolusi dari setiap *dataset* tersebut dengan tujuan agar tidak membutuhkan lebih banyak memori dan waktu pada saat proses pelatihan. Sementara, augmentasi data dapat meningkatkan jumlah *dataset* sehingga model yang dibuat dapat mengenali objek dalam berbagai situasi.
5. **Perancangan Sistem**
   Pada tahap ini, setelah melakukan *preprocessing* data, peneliti akan melakukan perancangan sistem dan mengimplementasikan metode yang akan digunakan ke sistem yang akan dibuat agar sistem dapat mendeteksi kendaraan, pelat, dan pengendara sepeda motor.
6. **Analisis Kinerja Sistem**
   Pada tahap ini, sistem yang telah dibuat akan diuji untuk mengevaluasi tingkat akurasi pada sistem. Hasil pengujian akan diselidiki secara menyeluruh untuk mengidentifikasi potensi permasalahan yang mungkin timbul dalam sistem e-tilang.
7. **Pembuatan Laporan**
   Pada tahap ini, proses penelitian yang telah dilakukan ditulis dalam bentuk skripsi dan ditarik kesimpulan dari hasil penelitian. Pada tahap pembuatan laporan ini, peneliti akan mengevaluasi hasil penelitian berdasarkan analisis sebelumnya.

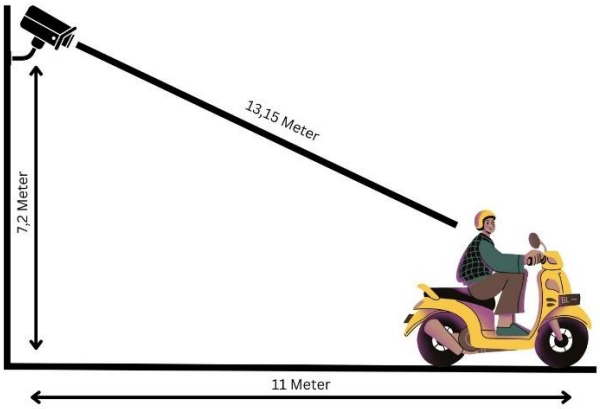
2.4 Teknik Pengambilan Data

Teknik pengambilan data dilakukan dengan menempatkan kamera di atas jembatan yang terlatak di Jalan Urip Sumoharjo dengan bantuan dari tripod kamera dengan tinggi 7,2 meter dan sudut 60°. Fokus utama pada penelitian ini adalah pada pengendara motor yang tidak menggunakan helm. Apabila terdapat pengendara tidak menggunakan helm pada data video yang diambil, maka sistem yang akan dibuat nantinya akan menangkap gambar pelat pengendara yang tidak menggunakan helm.

Pengambilan data pada penelitian ini berupa video dan dilakukan dari arah belakang kendaraan karena terdapat kondisi kendaraan dimana pengendara menggunakan helm sedangkan orang yang dibonceng oleh pengendara tidak menggunakan helm. Jadi, untuk dapat mendeteksi pembonceng tidak menggunakan helm pengambilan data harus dilakukan dari arah belakang.

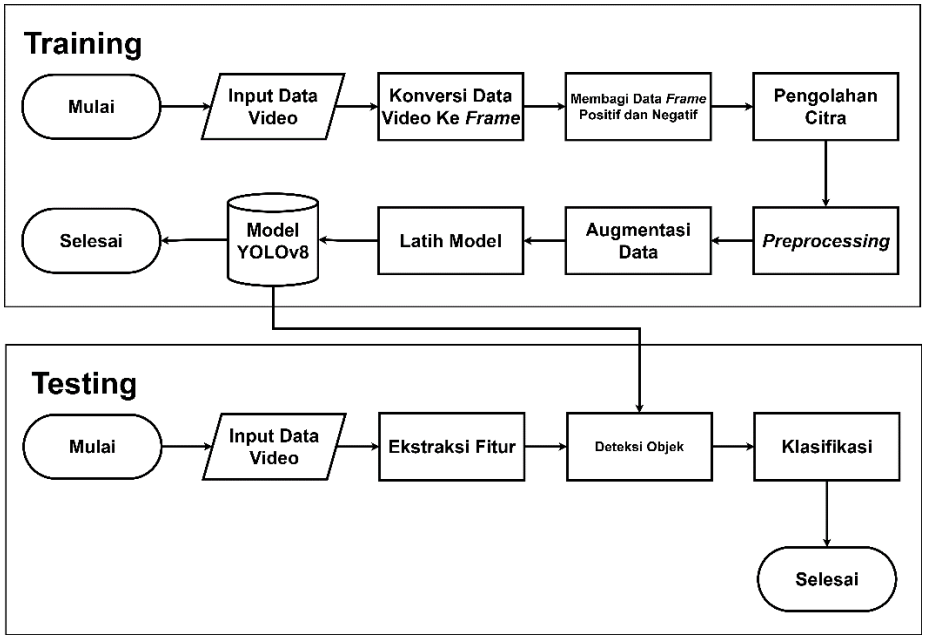
Pada penelitian ini digunakan kamera *handphone* dengan resolusi 1920x1080 dan *frame rate* 30 fps. Pemilihan resolusi ini karena data yang akan di *capture* ialah helm

untuk mengetahui pengendara yang tidak menggunakan helm atau tidak beserta dengan pelatnya. Adapun ilustrasi pengambilan data diperlihatkan pada Gambar 7.



Gambar 7. Deskripsi Posisi Kamera dan Kendaraan yang Dideteksi

2.5 Perancangan Sistem



Gambar 8. *Flowchart* Perancangan Sistem

Dalam perancangan sistem, untuk mendeteksi helm dan pelat pengendara sepeda motor terdapat beberapa tahapan sebagaimana yang ditunjukkan pada Gambar 8.

Sistem ini dibagi menjadi dua tahapan, yaitu tahapan *training* dan tahapan *testing*. Dalam tahapan *training*, hal pertama dilakukan adalah menginput data dengan format video (MP4) untuk melatih model deteksi objek yang akan dimasukkan ke dalam sistem. Setelah menginput data video, data tersebut akan dikonversi ke dalam bentuk *frame*. Hal ini bertujuan agar data yang telah dikonversi dapat melakukan *preprocessing*. Setelah

itu, data *frame* tersebut akan dilanjutkan ke tahap pelabelan gambar dan augmentasi data. Lalu, data yang telah dilabel dan diaugmentasi akan dilakukan tahap pelatihan model. Pada tahap ini, algoritma deteksi objek akan dipelajari oleh model dengan menggunakan data yang telah dimasukkan sebelumnya. Setelah model telah dilatih pada tahapan *training*, model tersebut akan digunakan pada tahapan *testing*.

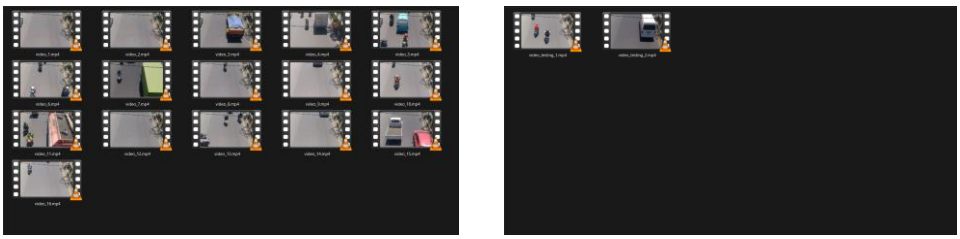
Tahapan *training* bertujuan untuk melatih sistem agar dapat mengenali data yang telah dilatih dengan kelompok klasifikasi yang telah ditentukan. Sedangkan tahapan *testing* bertujuan untuk mengetahui kinerja sistem. Data yang digunakan pada tahapan *training* dan *testing* menggunakan data yang berbeda.

1. **Input Data Video**

   Proses input data video merupakan langkah awal dalam perancangan sistem dari tahapan *training* dan *testing*. Adapun spesifikasi dari video adalah sebagai berikut:

   a. Resolusi : 1920x1080 piksel
   b. *Frame Rate* : 30 FPS
   c. Durasi : 60 detik (*training*) dan 5 detik (*testing*)

   Video input sistem merupakan hasil pengambilan video dengan menggunakan kamera statis yang disimpan dengan format video MP4. Total pengambilan video terdapat sebanyak 18 video. Dari 18 video tersebut akan dibagi menjadi 16 video untuk tahapan *training* dan 2 video untuk tahapan *testing*. Data video diambil langsung dari lokasi penelitian agar lebih relevan dengan situasi sebenarnya. Berikut ini adalah contoh data video yang telah diambil pada Gambar 9.



(a) (b)

Gambar 9. (a) Data Video untuk Tahapan *Training*, (b) Data Video untuk Tahapan *Testing*

2. **Konversi Data Video ke *Frame***

   Setelah melakukan pengumpulan data video, selanjutnya data video tersebut akan dikonversi menjadi *frame*. Data tersebut perlu dikonversi menjadi *frame* agar data *frame* yang telah dikumpulkan dapat digunakan pada tahap *preprocessing* nantinya. Untuk mengetahui jumlah data citra yang akan diperoleh dari konversi data video ke data *frame* dapat dilihat pada persamaan (11).

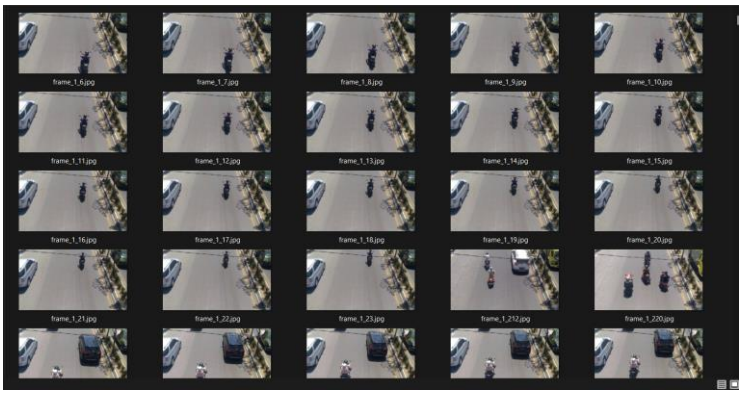
$$Total\ Frame = Durasi\ Video\ (Sec) \times Frame\ Rate \quad (11)$$

   Pada persamaan (11), dapat dilihat bahwa durasi video dari setiap data yang diperoleh adalah sekitar 60 detik. Karena diketahui *frame rate* dari kamera yang digunakan untuk mengambil data adalah 30 fps, sehingga total *frame* dalam 1

data video yang diambil adalah sekitar 18000 *frame*. Jadi, terdapat sekitar kurang dari 28800 *frame* hasil dari konversi dari 16 data video tersebut. Adapun potongan kodingan untuk mengonversi dari video ke *frame* dapat dilihat potongan *source code* berikut.

```
...
i = 0

for videos in glob.glob1(video_path, '*.mp4'):
    if (os.path.isdir(frames_path + '/video_' + str((i + 1))) !=
       True):
        os.mkdir(frames_path + '/video_' + str((i + 1)))

        jumlah_frame = len(glob.glob1(frames_path + '/video_' +
               str((i + 1)), "*.jpg"))

        cap = cv2.VideoCapture(video_path + "/" + videos)

        while True:
            ret, frame = cap.read()
            if not ret:
                break;
            cv2.imwrite(os.path.join(frames_path + '/video_' + str((i
                      + 1)), 'frame_' + str((i + 1)) + '_' +
                      str((jumlah_frame + 1)) + '.jpg'), frame)
            jumlah_frame += 1
    i += 1
```

Dari potongan kodingan tersebut akan menghasilkan *output* seperti pada Gambar 10 dan untuk melihat data tersebut secara lengkap dapat dilihat pada Lampiran 1.



Gambar 10. Contoh Data *Frame* yang Telah Dikonversi

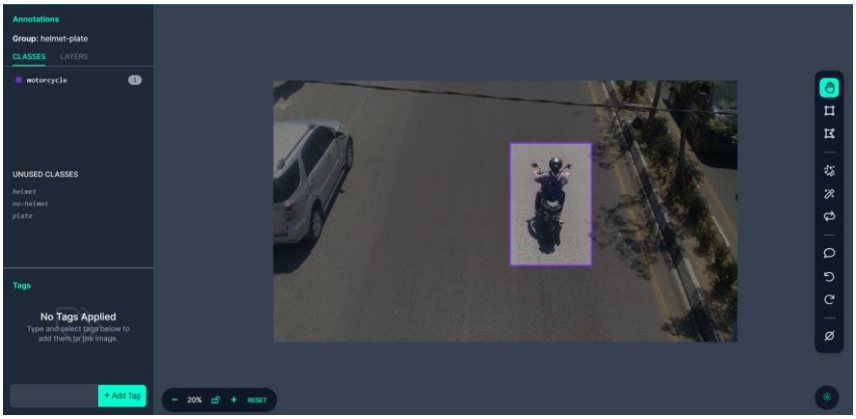
3. **Membagi Data *Frame* Positif dan Data *Frame* Negatif**
   Dari 28800 data *frame*, data tersebut akan dibagi menjadi dua bagian, yaitu data *frame* positif dan data *frame* negatif. data *frame* positif merupakan data *frame* yang memiliki objek kendaraan motor dari *frame* tersebut. Sedangkan data *frame* negatif merupakan data *frame* yang tidak memiliki objek kendaraan motor dari *frame* tersebut. Sehingga, jumlah total data untuk *frame* positif adalah 15392 *frame* dan untuk data *frame* negatif adalah 12779 *frame*. Adapun contoh data *frame* positif dan negatif dapat dilihat pada Gambar 11.

(a) (b)

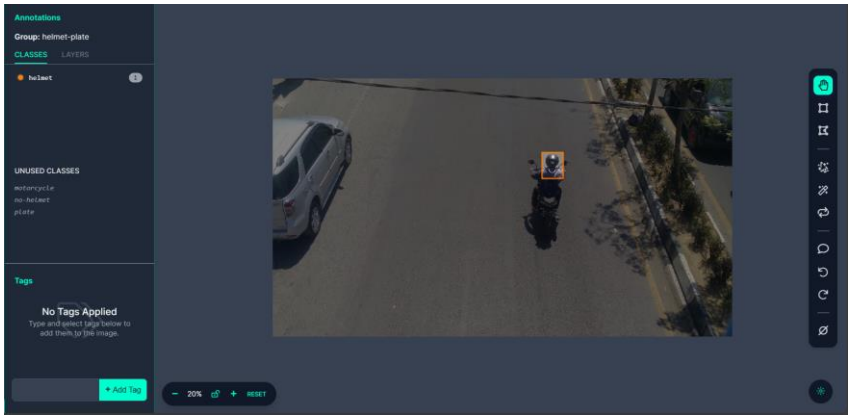
Gambar 11. (a) Contoh Data *Frame* Positif, (b) Contoh Data *Frame* Negatif

Data *frame* positif yang terkumpul sebanyak 15392 *frame* tersebut hanya akan digunakan sebanyak 1460 *frame* untuk *training* model nantinya.

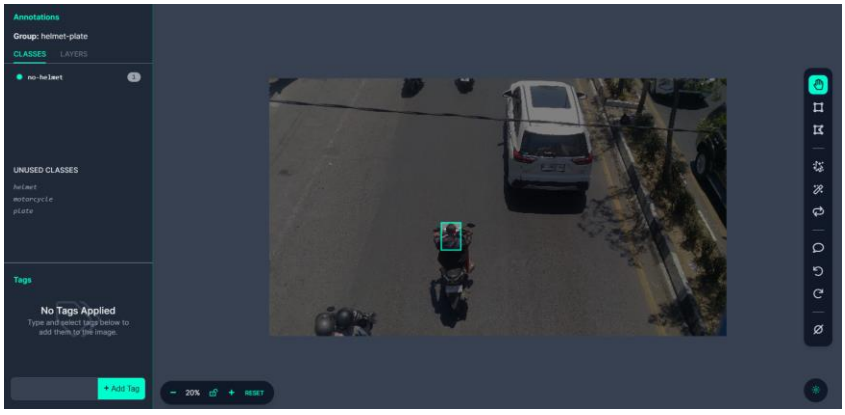
4. ***Labeling images***
   Setelah melakukan pembagian data, tahap selanjutnya adalah melakukan pelabelan (*labeling*) pada gambar. *Labeling images* adalah proses mengidentifikasi objek di dalam gambar dan menentukan batas area di sekitar objek tersebut, yang disebut sebagai *bounding box*. *Labeling images* sebelum proses augmentasi data bertujuan untuk menjaga konsistensi label sesuai dengan objek yang sebenarnya dalam gambar, serta untuk mencegah kemungkinan konflik antara label dan data yang telah diubah, yang mungkin terjadi jika penandaan gambar dilakukan setelah augmentasi data. Setiap gambar dalam *dataset* harus dilengkapi dengan informasi *bounding box* yang menandai lokasi objek di dalamnya, bersama dengan label kelas yang sesuai.
   Sebelum melakukan proses pelabelan gambar, perlu diketahui bahwa pelabelan data ini dilakukan dengan bantuan Roboflow dan terdapat empat jenis objek atau kelas yang akan diberi label pada penelitian ini. Yang pertama adalah kendaraan sepeda motor (Gambar 12), yang digunakan untuk mengetahui keberadaan pengendara sepeda motor dari gambar tersebut. Yang kedua adalah pengendara sepeda motor yang memakai helm (Gambar 13), yang digunakan untuk mengetahui pengendara yang menggunakan helm. Yang ketiga adalah pengendara yang tidak menggunakan helm (Gambar 14), yang digunakan untuk mengetahui pengendara sepeda motor yang tidak menggunakan helm. Yang terakhir adalah pelat kendaraan (Gambar 15), yang digunakan untuk mengetahui pelat kendaraan sepeda motor. Adapun contoh pelabelan data sebagai berikut.



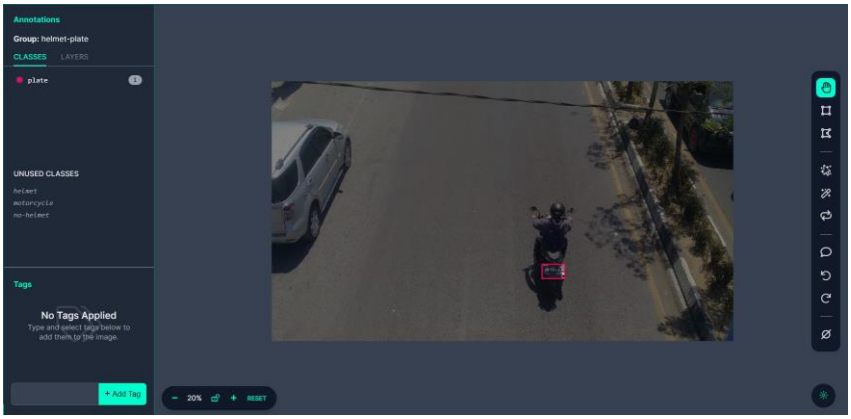
Gambar 12. Contoh Label Kendaraan Sepeda Motor



Gambar 13. Contoh Label Pengendara Mengenakan Helm



Gambar 14. Contoh Label Pengendara Tidak Mengenakan Helm



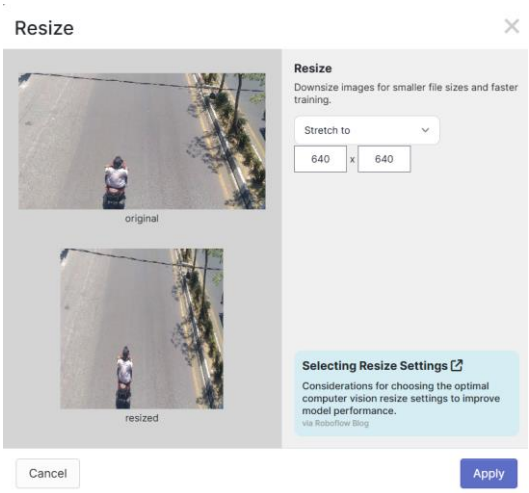
Gambar 15. Contoh Label Pelat Kendaraan Motor

5. *Preprocessing*

Proses selanjutnya adalah melakukan tahapan *preprocessing*. Tahapan *preprocessing* dalam penelitian ini adalah *image resize*. *Resize* dalam deteksi objek adalah langkah mengubah ukuran gambar atau citra agar sesuai dengan kebutuhan input model. Tindakan ini tidak hanya mengoptimalkan penggunaan memori dan sumber daya komputasi, tetapi juga memberikan kendali terhadap proporsi gambar. Dengan menyesuaikan ukuran gambar, hal ini juga dapat meningkatkan kemampuan model dalam mendeteksi objek kecil atau detail halus, serta mempermudah perbandingan objek dengan ukuran yang seragam.

Resize dilakukan dengan memodifikasi ukuran tinggi dan lebar gambar, menjaga konsistensi aspek rasio, dan konten visualnya. Ukuran 640x640 piksel dipilih karena dianggap optimal dan sesuai untuk pengolahan dalam gambar. Tahap *resize* dalam *preprocessing* data gambar memegang peranan penting dalam meningkatkan penggunaan YOLOv8. Ukuran input yang tetap dan sesuai dengan kebutuhan YOLOv8 membantu meningkatkan kinerja model serta mempermudah implementasi dan penggunaan algoritma deteksi objek ini dalam berbagai aplikasi yang berkaitan dengan *vision-based tasks*.

Proses dalam tahapan *resize image* akan dilakukan pada platform Roboflow Ketika seluruh *frame* atau citra telah dimasukkan ke dalam Roboflow. Hal ini dikarenakan Roboflow akan sangat membantu pengguna dalam mengolah data citra salah satunya adalah dalam *resizing image*. Adapun hasil *resize image* dapat dilihat pada Gambar 16.



Gambar 16. Proses *Resize Images*

6. Augmentasi Data

Setelah data telah melakukan tahap *preprocessing*, selanjutnya data tersebut akan dilakukan augmentasi data. Augmentasi data merupakan teknik yang umum digunakan untuk meningkatkan kinerja model dengan cara memperluas variasi data yang tersedia. Salah satu tujuan utama dari augmentasi data adalah untuk meningkatkan kinerja model. Dengan memperluas variasi data yang tersedia untuk pelatihan, augmentasi data memungkinkan model untuk belajar pola yang lebih umum dan kompleks. Hal ini dapat menghasilkan model yang lebih baik dalam melakukan tugas deteksi objek dengan lebih akurat dan konsisten.

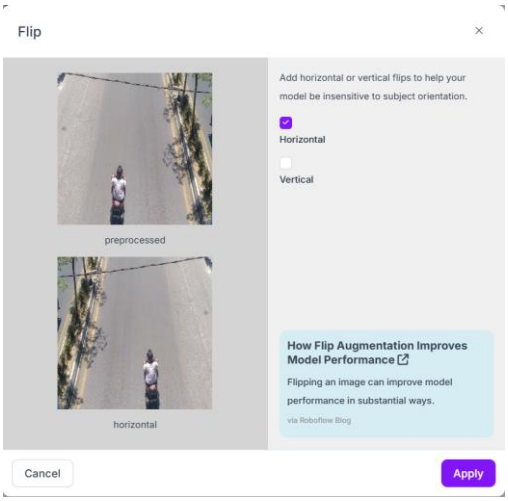
Selain itu, augmentasi data juga membantu mengatasi masalah *overfitting*. *Overfitting* terjadi ketika model terlalu spesifik terhadap data pelatihan dan gagal untuk menggeneralisasi pola ke data baru. Dengan memperkenalkan variasi tambahan melalui augmentasi data, model lebih cenderung belajar fitur-fitur yang lebih umum dan tidak hanya mengandalkan pola yang terdapat dalam data pelatihan saja. Masalah keterbatasan data juga dapat diatasi dengan menggunakan augmentasi data. Dalam banyak kasus, jumlah data yang tersedia untuk pelatihan model mungkin terbatas. Augmentasi data juga dapat "menggandakan" atau memperbanyak *dataset* pelatihan dengan membuat variasi baru dari data yang ada, sehingga membuat model memiliki lebih banyak contoh untuk belajar.

Adapun augmentasi data yang digunakan pada penelitian ini adalah:

a. *Flip Images*

Flip images merupakan salah satu teknik augmentasi data yang memberikan variasi tambahan dalam *dataset* pelatihan. Dengan memasukkan versi terbalik dari gambar-gambar tersebut, model *machine learning* dapat belajar untuk mengenali objek dalam berbagai orientasi. Terdapat dua jenis dalam melakukan *flip images*, yaitu pembalikan citra secara horizontal dan vertikal.

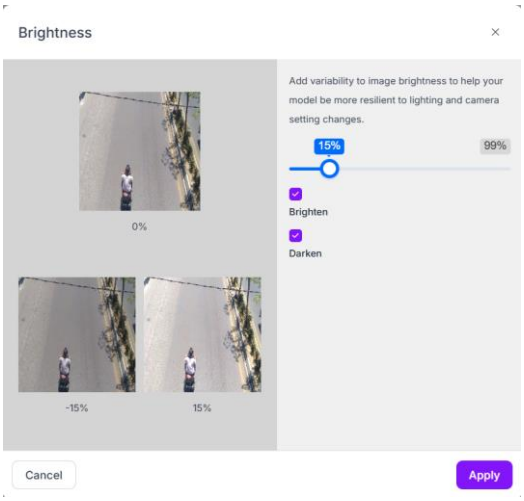
Akan tetapi, pada penelitian ini hanya menggunakan pembalikan citra secara horizontal. Adapun hasil *flip images* dapat dilihat pada Gambar 17.



Gambar 17. Proses *Flip Images*

b. *Brightness*

Brightness adalah teknik dalam pelatihan model pembelajaran mesin yang mengubah kecerahan gambar untuk meningkatkan variasi data dan membuat model lebih tahan terhadap kondisi pencahayaan yang berbeda. Teknik ini membantu mengurangi *overfitting* dengan memungkinkan model belajar dari berbagai kondisi pencahayaan tanpa perlu mengumpulkan data tambahan. Kecerahan gambar dari penelitian ini menurunkan dan menaikkan pencahayaan sebanyak 15% dari gambar aslinya. Adapun hasil *brightness* dapat dilihat pada Gambar 18.



Gambar 18. Proses *Brightness*

Pada penelitian ini, Proses augmentasi data akan dilakukan dengan 3 skenario yang berbeda-beda. Adapun ketiga skenario diuraikan sebagai berikut.

Skenario 1

Pada skenario pertama, penelitian ini tidak akan diberikan proses augmentasi data sehingga jumlah data yang akan dilatih lebih sedikit.

Skenario 2

Pada skenario kedua, penelitian ini akan menggunakan 1 proses augmentasi data. Proses augmentasi data yang akan digunakan pada skenario kedua adalah augmentasi data *flip images*.

Skenario 3

Pada skenario ketiga, penelitian ini akan menggunakan 2 proses augmentasi data. Proses augmentasi data yang akan digunakan pada skeanrio ketiga adalah augmentasi data *flip images* dan *brightness*.

7. Latih Model (*Training Model*)

Setelah melalui proses augmentasi data, *dataset* tersebut akan dibagi menjadi 3 kelompok, yaitu:

Data *Training*: Data ini akan digunakan untuk melatih model atau algoritma yang akan diterapkan dalam proses analisis.

Data *Validation*: Data ini akan digunakan untuk mengukur seberapa baik model dapat melakukan deteksi objek pada data yang tidak terlihat selama pelatihan.

Data *Testing*: Data ini akan digunakan untuk menguji model untuk mengukur kinerja akhirnya secara objektif. Hasil pengujian pada data testing memberikan gambaran seberapa baik model akan bekerja dalam situasi dunia nyata dan membantu dalam menentukan apakah model siap untuk digunakan secara praktis.

Setelah membagi *dataset*, selanjutnya adalah *dataset* tersebut sudah bisa melalui proses tahapan pelatihan model dengan menggunakan algoritma YOLOv8. Model YOLOv8 harus dikonfigurasi dengan arsitektur jaringan yang sesuai dan parameter-parameter pelatihan yang optimal. Pilihan arsitektur jaringan, hyperparameter, dan metode optimisasi akan mempengaruhi kinerja model. Bobot awal mungkin diinisialisasi secara acak atau menggunakan bobot dari model *pretrained*.

Dalam peningkatan Model Algoritma YOLOv8, diperlukan pengoptimalan dengan bantuan *Stochastic Gradient Descent* (SGD) dengan nilai *learning rate* (*lr*) sebesar 0.01. Tahapan kunci dalam pengembangan model ini adalah melalui proses *training data*, yang bertujuan untuk memastikan model dapat mengenali *dataset* yang digunakan. Setelah melalui pelatihan, model yang terbentuk akan diuji menggunakan data *testing*. Adapun total *dataset* yang digunakan pada penelitian ini adalah sebanyak 1460 *dataset*.

Pada penelitian ini, dilakukan percobaan dengan menggunakan parameter sebagai berikut:

a. *Epoch* = 35

Epoch mengacu pada satu kali proses melatih model menggunakan seluruh set data pelatihan. Pada setiap *epoch*, model akan melewati setiap sampel pelatihan satu kali, menghitung *loss* antara prediksi yang dihasilkan oleh model dan label yang seharusnya. Dalam proses pelatihan pada penelitian ini, dilakukan sebanyak 35 *epochs*, di mana model akan menyesuaikan bobotnya sebanyak 35 kali. Peningkatan jumlah *epochs* ini memiliki dampak yang baik pada kemampuan model untuk mengenali objek dalam gambar. Namun, perlu dicatat bahwa meningkatkan jumlah epoch juga dapat memiliki beberapa dampak negatif, seperti meningkatnya waktu pelatihan dan risiko *overfitting* jika tidak diatasi dengan benar.

b. *Batch* = 16

Batch mengacu pada sejumlah sampel data yang digunakan secara bersamaan dalam satu iterasi pelatihan untuk menghitung *loss* dan memperbarui parameter model. Proses pelatihan menggunakan batch untuk meningkatkan efisiensi dan stabilitas pelatihan. Dalam implementasi YOLOv8, *batch size* yang digunakan sebanyak 16 *batch*, yang artinya model akan menangani 16 gambar sekaligus sebelum melakukan perubahan bobot. Pengaturan ini berdampak pada kecepatan pelatihan dan penggunaan memori. Namun, penting untuk memperhatikan bahwa pemilihan ukuran *batch* yang optimal juga memerlukan pertimbangan. Oleh karena itu, pemilihan ukuran *batch* biasanya melibatkan eksperimen dan penyetelan pada data validasi untuk menemukan ukuran *batch* yang optimal untuk model dan data yang spesifik.

c. *Image Size* (*imgsz*) = 800

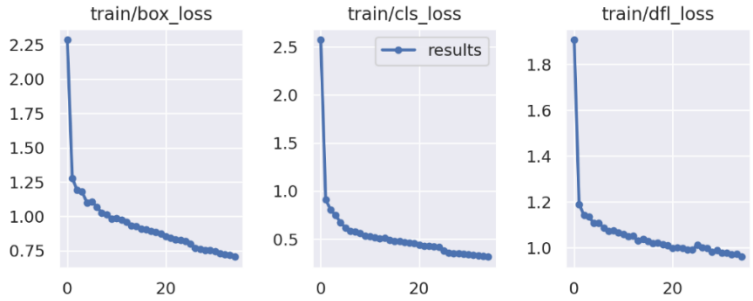
Image size mengacu pada dimensi spasial (panjang dan lebar) dari gambar input yang digunakan dalam pelatihan dan inferensi model. Ukuran gambar input ini memengaruhi bagaimana model YOLOv8 memproses dan menghasilkan prediksi objek. Dalam penelitian ini, peneliti menggunakan ukuran gambar 800x800 piksel. Ukuran gambar yang digunakan dapat memengaruhi kinerja model dalam mendeteksi objek dalam gambar. Semakin besar ukuran gambar, semakin banyak detail yang dapat diakses oleh model sehingga proses pelatihannya akan semakin lama dan begitupun sebaliknya.

Dalam tahap pelatihan model, gambar-gambar dari *dataset* pelatihan dimasukkan ke dalam model. Model kemudian melakukan prediksi terhadap gambar-gambar tersebut dan menghasilkan prediksi *bounding box* dan kelas objek. Fungsi kerugian (*loss function*) dihitung untuk membandingkan prediksi model dengan label yang sebenarnya, dan parameter model diperbarui menggunakan teknik *backpropagation*. Proses ini berulang kali dijalankan melalui seluruh *dataset* pelatihan (beberapa kali iterasi atau *epoch*) untuk mengoptimalkan model.

Berikut ini adalah hasil dari nilai *loss function* dari proses pelatihan dari ketiga skenario yang telah dilakukan:

a. Skenario 1: Tanpa Proses Augmentasi Data

Dari skenario pertama, berdasarkan hasil *training* yang dilakukan, yang dimana jumlah *dataset* apabila tanpa melakukan proses augmentasi data adalah 1460 data. Data tersebut akan dibagi menjadi 3 bagian, yang pertama adalah data *training* dengan total 940 data (64%), kedua yaitu data validasi dengan total 374 data (26%), dan yang ketiga adalah data *testing* dengan total 146 data (10%). Berdasarkan hasil *training* yang telah dilakukan memiliki nilai *loss* yang dihasilkan selama proses *training* berjalan. Adapun grafik *loss* yang dihasilkan selama proses *training* dapat dilihat pada Gambar 19.



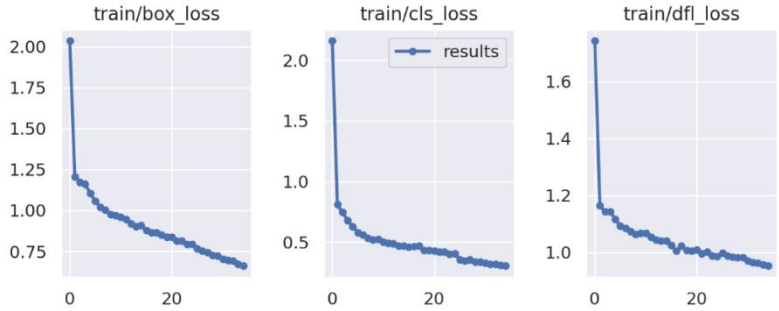
Gambar 19. Grafik *Loss* dari Hasil *Training* Model Skenario 1

Berdasarkan grafik nilai *loss* yang ditampilkan pada Gambar 19, terlihat adanya penurunan nilai pada proses *training* model pada skenario 1 baik dari grafik *training*. Dari hasil tersebut menunjukkan bahwa proses *training* model berjalan dengan baik dan efektif. Akan tetapi, untuk mengukur apakah model telah dilatih dengan baik dari proses *training* itu perlu menghitung keakuratan dari proses *testing*.

Untuk *box loss training* dapat dilihat bahwa nilai tersebut menurun dari sekitar 2,2 menjadi 0,7. Untuk *classification loss training* dapat dilihat bahwa nilai tersebut menurun dari sekitar 2,5 menjadi 0.3. Berdasarkan hasil tersebut, terlihat bahwa nilai akhir dari *loss training* sudah mendekati nilai 0 yang menandakan model telah dilatih dengan baik dari proses *training*.

b. Skenario 2: Dengan 1 Proses Augmentasi Data (*Flip Images*)

Dari skenario kedua, berdasarkan hasil *training* yang dilakukan, yang dimana jumlah *dataset* apabila dengan melakukan proses augmentasi data yaitu *flip images*, jumlah *dataset* akan bertambah dari 1460 menjadi 2108 data. Data tersebut akan dibagi menjadi 3 bagian, yang pertama adalah data *training* dengan total 1489 data (71%), kedua yaitu data validasi dengan total 390 data (19%), dan yang ketiga adalah data *testing* dengan total 229 data (11%). Berdasarkan hasil *training* yang telah dilakukan memiliki nilai *loss* yang dihasilkan selama proses *training* berjalan. Adapun grafik *loss* yang dihasilkan selama proses *training* dapat dilihat pada Gambar 20.

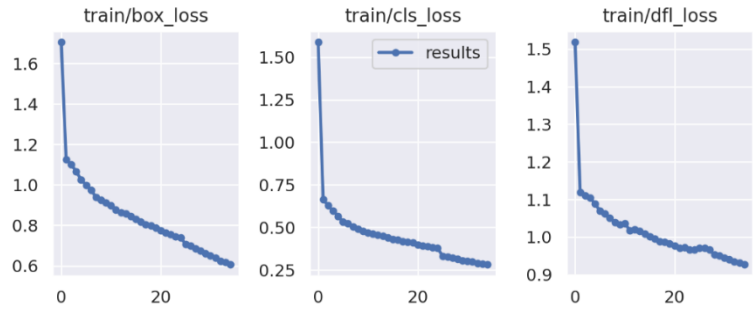


Gambar 20. Grafik *Loss* dari Hasil *Training* Model Skenario 2

Berdasarkan grafik nilai *loss* yang ditampilkan pada **Error! Reference source not found.**, terlihat bahwa nilai *box loss training* menurun dari sekitar 2 menjadi 0,6. Untuk *classification loss training* dapat dilihat bahwa nilai tersebut menurun dari sekitar 2,1 menjadi 0.3. Berdasarkan hasil tersebut, terlihat bahwa nilai akhir dari *loss training* sudah mulai mendekati nilai 0 yang menandakan model telah dilatih dengan baik dari proses *training*.

c. Skenario 3: Dengan 2 Proses Augmentasi Data (*Flip Images* & *Brightness*)

Dari hasil *training* yang dilakukan, yang dimana jumlah *dataset* apabila dengan melakukan proses augmentasi data yaitu *flip images* dan *brightness*, jumlah *dataset* akan bertambah dari 1460 menjadi 3479 data. Data tersebut akan dibagi menjadi 3 bagian, yang pertama adalah data *training* dengan total 3033 data (87%), kedua yaitu data validasi dengan total 278 data (8%), dan yang ketiga adalah data *testing* dengan total 168 data (5%). Berdasarkan hasil *training* yang telah dilakukan memiliki nilai *loss* yang dihasilkan selama proses *training* berjalan. Adapun grafik *loss* yang dihasilkan selama proses *training* dapat dilihat pada Gambar 21.



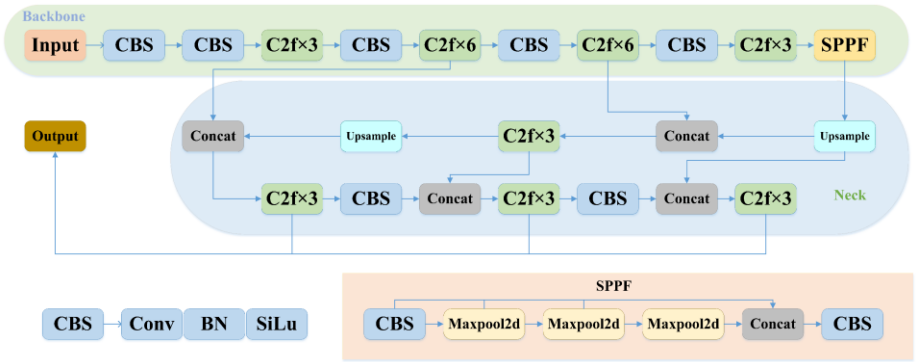
Gambar 21. Grafik *Loss* dari Hasil *Training* Model Skenario 3

Berdasarkan grafik nilai *loss* yang ditampilkan pada **Error! Reference source not found.**, terlihat bahwa nilai *box loss training* menurun dari sekitar 1,7 menjadi 0,6. Untuk *classification loss training* dapat dilihat bahwa nilai tersebut menurun dari sekitar 1,58 menjadi 0.28. Berdasarkan hasil tersebut, terlihat bahwa nilai akhir dari *loss training* sudah mulai mendekati nilai 0 yang menandakan model telah dilatih dengan baik dari proses *training*.

8. *Feature Extraction*, Deteksi Objek, dan *Classification*

Setelah melalui proses tahapan *training*, selanjutnya model yang telah dilatih akan diuji untuk mengukur akurasi dan ketingkatan model dalam mengidentifikasi objek yang benar. Model diuji menggunakan *dataset* yang telah melalui proses pelabelan dan augmentasi data dengan informasi tentang lokasi dan kelas objek yang sebenarnya.

Pada tahapan *testing*, terdapat proses ekstraksi fitur atau *feature extraction* yang merupakan proses mengambil informasi penting dari data yang kompleks. Dalam konteks deteksi objek, data yang kompleks ini adalah gambar. Gambar dapat memiliki jutaan piksel dengan nilai intensitas yang berbeda di setiap pikselnya. Ketika model ingin mendeteksi objek di dalam gambar, model tidak perlu mempertahankan semua informasi piksel tersebut. Sebaliknya, model akan mengekstrak fitur-fitur kunci yang mewakili objek yang akan dideteksi. Adapun proses *feature extraction* dapat dilihat pada Gambar 22.



Gambar 22. Proses Ekstraksi Fitur pada YOLOv8

a. *Backbone*

Proses *backbone* merupakan bagian utama dari arsitektur jaringan yang bertanggung jawab atas ekstraksi fitur-fitur penting dari gambar input. *Backbone* merupakan langkah awal dalam proses deteksi objek, di mana gambar diolah melalui beberapa lapisan konvolusi untuk memperoleh representasi yang lebih abstrak. Berikut ini adalah lapian dari proses *backbone*:

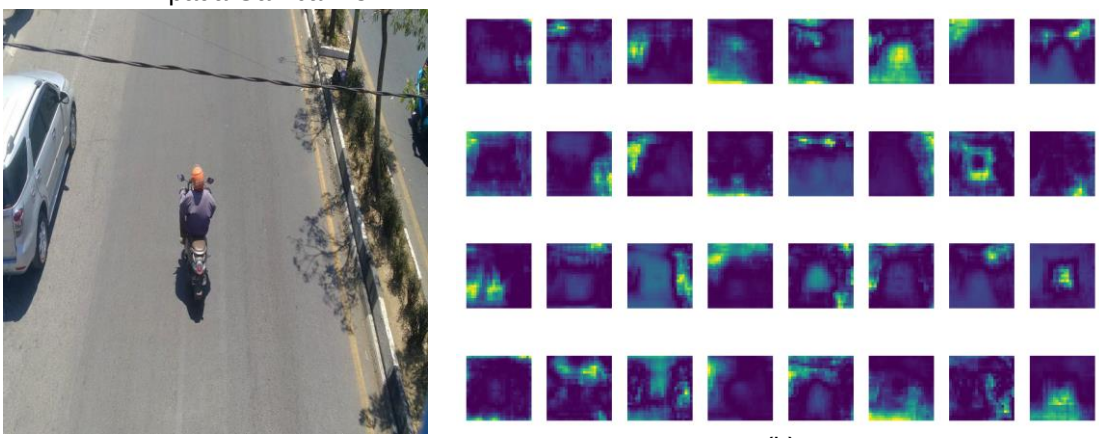
CBS adalah singkatan dari *Conv-BN-SiLU*, yang merujuk pada urutan tiga lapisan yaitu *Convolutional* yang digunakan untuk mengekstrak fitur dasar dari gambar, *Batch Normalization* yang digunakan untuk menstabilkan pelatihan dengan menormalisasi output konvolusi, dan *Sigmoid Linear Unit* yang digunakan untuk memberikan kemampuan bagi model untuk mengenali pola yang lebih kompleks.

C2f adalah singkatan dari CSP (*Cross-Stage Partial*) + 2 *convolutions* dan 1 *Feedforward* yang digunakan untuk meningkatkan efisiensi ekstraksi fitur dan mengurangi beban komputasi.

SPPF adalah singkatan dari *Spatial Pyramid Pooling – Fast* yang digunakan untuk mempercepat ekstraksi fitur dan meningkatkan kinerja

deteksi objek, terutama dalam menangani gambar dengan ukuran berbeda-beda.

Berikut ini adalah hasil ekstraksi fitur pada proses *backbone* dapat dilihat pada Gambar 23.



(a) (b)

Gambar 23. (a) Contoh Gambar yang Diinput, (b) Ekstraksi Fitur pada Proses *Backbone*

b. *Neck*

Proses *neck* dalam arsitektur YOLOv8 merujuk pada tahap-tahap yang terjadi setelah ekstraksi fitur dari lapisan-lapisan konvolusi dan sebelum *output* deteksi objek. Tahap-tahap ini penting untuk memperoleh representasi fitur yang lebih abstrak dan kaya sehingga memungkinkan model untuk lebih akurat dalam mendeteksi objek. Terdapat lapisan tambahan dalam proses yang terjadi pada proses *neck*, yaitu:

Upsample adalah proses yang digunakan untuk meningkatkan resolusi fitur yang telah diekstraksi, sehingga bisa lebih mudah digabungkan dengan fitur dari skala yang lebih besar.

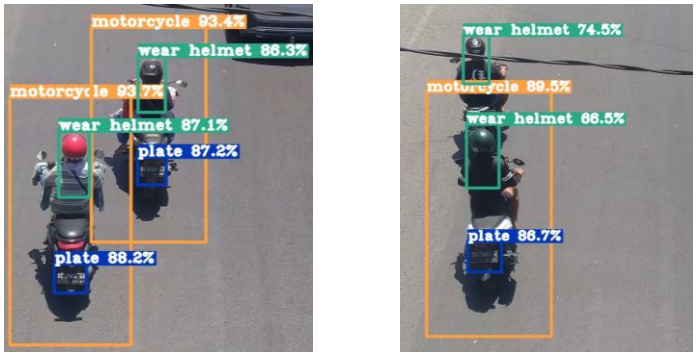
Concat adalah proses yang digunakan untuk menggabungkan informasi fitur dari berbagai level skala yang telah diproses oleh berbagai lapisan, seperti hasil dari *upsampling* atau fitur dari berbagai tahap dalam backbone.

c. *Head*

Pada proses *head* dalam arsitektur YOLOv8 menggunakan struktur *decoupled head* yang dimana proses *head* akan dipisahkan menjadi *object classification* dan prediksi regresi *bounding box*. *Object classification* berfungsi untuk mengelompokkan objek berdasarkan kategori atau label yang spesifik dan menggunakan *binary cross-entropy loss* (BCE *Loss*) yang dikenal sebagai kerugian log atau kerugian logistik, umumnya digunakan dalam tugas klasifikasi biner. Sedangkan prediksi regresi *bounding box* menggunakan *Distribution Focal Loss* (DFL) dan CIoU untuk meningkatkan kualitas deteksi objek dalam skenario di mana distribusi ukuran objek sangat bervariasi. Hal ini membantu model YOLOv8 untuk lebih efektif dalam

mengatasi masalah ketidakseimbangan distribusi ukuran objek dalam *dataset*, sehingga dapat meningkatkan akurasi deteksi objek pada objek-objek yang kecil atau besar.

Permasalahan yang sering muncul pada deteksi objek adalah kesulitan dalam mendeteksi objek yang saling bertumpukan. Hal ini disebabkan oleh keterbatasan kemampuan model dalam membedakan batas objek yang saling tumpang tindih, terutama jika objek tersebut memiliki ukuran, bentuk, atau tekstur yang serupa. Model yang tidak mampu menangani kasus ini dengan baik dapat menghasilkan deteksi yang gagal mengenali objek yang tersembunyi sebagian di balik objek lain. Tetapi, jika objek masih saling berdekatan dengan objek yang lain, maka model masih mampu mendeteksi kedua objek tersebut. Adapun contoh objek yang berdekatan dan bertumpukan dapat dilihat pada Gambar 24.



(a) (b)

Gambar 24. (a) Contoh Kendaraan yang Berdekatan dengan Kendaraan Lain, (b) Contoh Kendaraan yang Bertumpukan dengan Kendaraan Lain

Setelah melalui deteksi objek, proses yang terakhir adalah melakukan proses klasifikasi pada deteksi objek di mana YOLOv8 memiliki peran penting dalam memastikan bahwa objek yang terdeteksi tidak hanya dikenali sebagai adanya objek, tetapi juga dikategorikan dengan benar ke dalam kelas yang sesuai. Adapun contoh hasil *output* pada penelitian ini dapat dilihat pada Gambar 25.

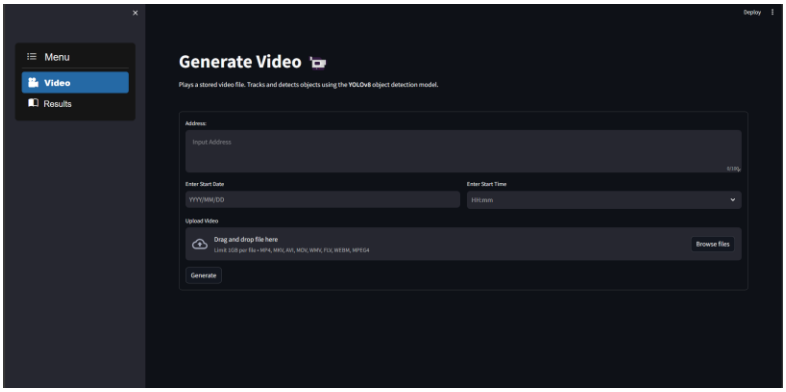
(a) (b)

Gambar 25. (a) Contoh Gambar yang Dideteksi, (b) *Output* Hasil Deteksi Objek

2.6 Implementasi Program

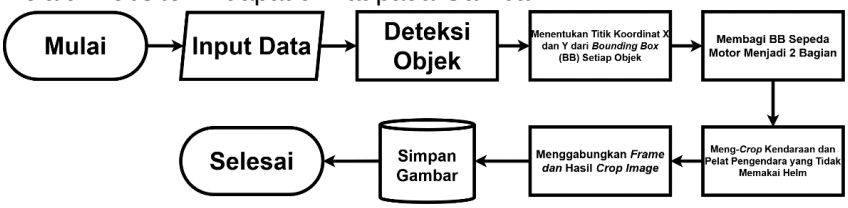
Sistem yang dibuat adalah sebuah *interface* berbasis *website* sederhana dengan menggunakan *framework Streamlit*. *Streamlit* adalah sebuah *framework open-source* yang digunakan untuk membangun aplikasi web interaktif dengan mudah menggunakan Python. Dengan adanya *framework* ini dapat membuat aplikasi web yang memanfaatkan kode Python untuk menghasilkan tampilan interaktif secara langsung, tanpa memerlukan penulisan HTML, CSS, atau JavaScript secara eksplisit. Adapun fitur dari *website* yang dibuat pada penelitian ini adalah:

1. **Input Data**



Gambar 26. Tampilan Awal *Website*

Dapat dilihat pada Gambar 26, fitur ini berisi inputan data berupa alamat (*address*), tanggal dan waktu pengambilan data, dan inputan *file* yang berupa video. Selain itu, terdapat juga tombol *generate* yang digunakan untuk melalui proses pendeteksian objek dengan menggunakan model yang telah melalui tahapan *training* sebelumnya untuk menghasilkan *output* yang akan diimplementasikan pada menu *results*. Adapun proses sistem yang akan dibuat melalui *website* ini dapat dilihat pada Gambar 27.



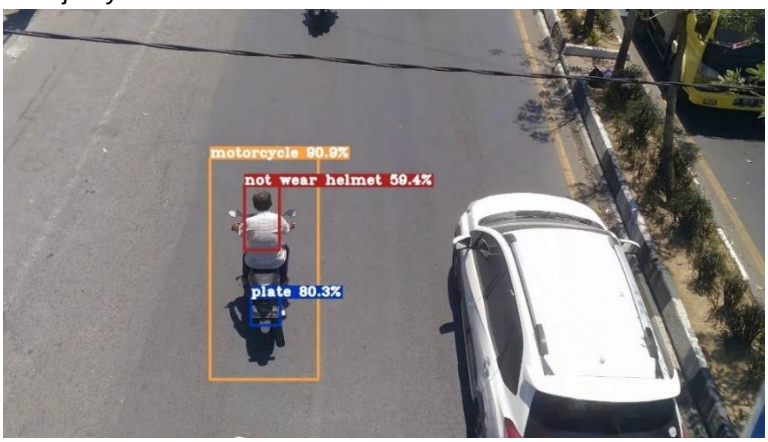
Gambar 27. Proses Sistem pada *Website* yang Dibuat

Gambar 27 memperlihatkan proses sistem dengan menggunakan model dari hasil tahapan *training* yang kemudian akan diimplementasikan melalui aplikasi berbasis *website*. Adapun prosesnya adalah:

Step 1: Menginput data atau *file* berupa video dari inputan *file* yang telah disediakan oleh *website* yang telah dibuat (Gambar 26).

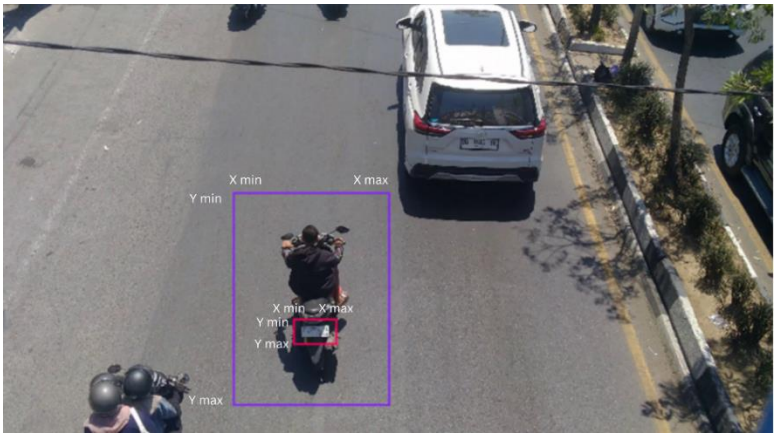
Step 2: Setelah itu, sistem akan mendeteksi tiap *frame* dari video yang dimasukkan dengan menggunakan model hasil dari tahapan *training* yang

telah dibuat. Apabila terdapat pengendara yang tidak mengenakan helm pada *frame* tersebut, maka sistem akan melakukan pada tahapan atau *step* selanjutnya. Perhatikan Gambar 28.



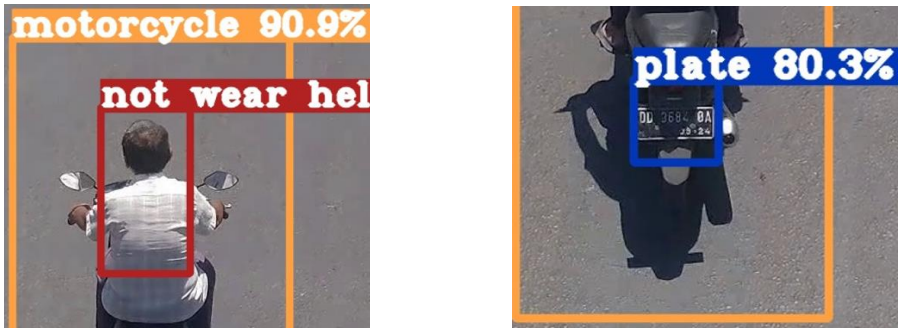
Gambar 28. Contoh *Frame* Hasil Deteksi Objek

Step 3: Setelah itu, kegunaan *bounding box* pada objek kendaraan dan pelat sepeda motor digunakan untuk mendapatkan titik koordinat *X* dan *Y* seperti yang ditampilkan pada Gambar 29 yang nantinya akan digunakan untuk meng-*crop* kendaraan maupun pelat sepeda motor.



Gambar 29. Titik Koordinat *X* dan *Y* dari Label Kendaraan dan Pelat Sepeda Motor

Step 4: Setelah mendapatkan koordinat *X* dan *Y* dari hasil deteksi objek, selanjutnya adalah *bounding box* dari kendaraan sepeda motor tersebut akan dibagi menjadi 2 bagian seperti yang ditampilkan pada Gambar 30.



(a) (b)

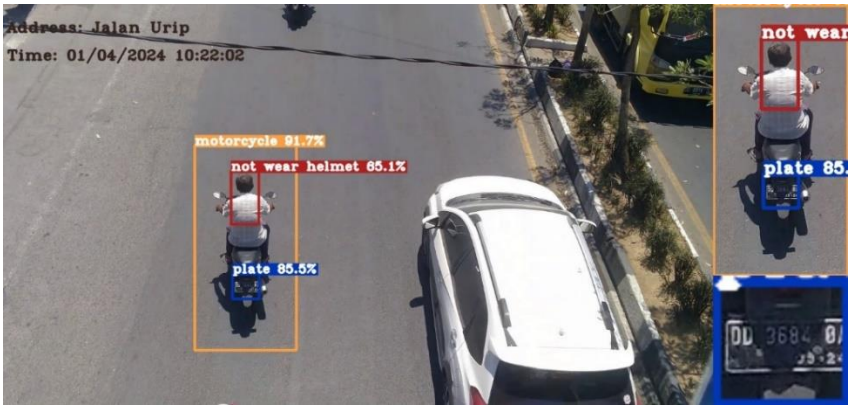
Gambar 30. (a) *Bounding Box* Kendaraan Sepeda Motor Bagian Atas, (b) *Bounding Box* Kendaraan Sepeda Motor Bagian Bawah

Pada Gambar 30, dapat dilihat bahwa kegunaan *bounding box* kendaraan sepeda motor dibagi menjadi dua adalah untuk dapat membedakan bagian atas dan bawah kendaraan sepeda motor. Untuk bagian atas kendaraan sepeda motor terdapat pengendara sepeda motor, sehingga sistem dapat mecari tahu apakah pengendara sedang memakai helm atau tidak. Sedangkan, untuk bagian bawah kendaraan sepeda motor terdapat pelat kendaraan, sehingga sistem nantinya dapat meng-*crop* pelat tersebut.

Bounding box kendaraan sepeda motor tersebut perlu dibagi menjadi dua agar pada saat sistem dapat mengetahui si pengendara atau pelat dari kendaraan tersebut. Jangan sampai terjadi kesalahan saat mendeteksi pengendara atau pelat kendaraan tersebut dikarenakan kendaraan sepeda motor tersebut saling berhimpitan dengan kendaraan yang lain. Sebagai contoh dapat dilihat pada Gambar 31.

Gambar 31. Contoh Kondisi Dimana Kendaraan yang Berdekatan dengan Kendaraan Lain

Step 5: Setelah *bounding box* kendaraan sepeda motor telah dibagi menjadi dua bagian, selanjutnya adalah meng-*crop* kendaraan sepeda motor beserta dengan pelatnya dari pengendara yang tidak menggunakan helm tersebut, lalu kemudian menggabungkan hasil *crop* tersebut dengan *frame* yang di *crop* sehingga hasilnya seperti pada Gambar 32.



Gambar 32. *Output* dari Sistem *Website* yang Telah Dibuat

Setelah itu, *output* tersebut akan disimpan ke dalam *database MySQL* lalu ditampilkan pada menu '*Results*'.

2. Hasil *Generate* Video

Gambar 33. Tampilan Menu '*Results*'

Pada Gambar 33, dapat dilihat bahwa gambar tersebut menampilakn hasil *output* berupa gambar dan video dari sistem yang telah dibuat yang dimana hasil tersebut menampilakan pengendara yang tidak menggunakan helm.